Finding Good Counter-Examples to Aid Design Verification

Görschwin Fey Rolf Drechsler
Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
{fey,drechsle}@informatik.uni-bremen.de

Abstract

Today up to 80% of the design costs for integrated circuits are due to verification. Verification tools guarantee completeness if equivalence of two designs or a property for a design are proven. In the other case usually only one counter-example is produced. Then debugging has to be carried out to locate the design error.

This paper investigates, how debugging can benefit from using more than one counter-example generated by the verification tool. The problem of finding useful counter-examples is theoretically analyzed and proven to be difficult. Heuristics are introduced and their quality is underlined by experimental results. Guidelines how to generate counter-examples are extracted from one of these heuristics.

1. Introduction

The design gap and especially the verification gap is increasing. Circuits of several million gates can be produced but ensuring correct functional behavior becomes more and more challenging. Formal verification aids this task by guaranteeing equivalence of designs or validity of a property under any input sequence (see e.g. [3]). But the opposite, i.e. proving inequivalence of two designs or invalidity of a property is usually only done by providing one counter-example. This is formally correct, but the designer has to carry out the tedious task of locating the error based upon this one counter-example by hand.

In *Design Error Detection and Correction* (DEDC) frequently a set of test-vectors is used to reduce the number of candidate error sites (e.g. [4]). But this set of test-vectors is assumed to be given by a tool previous to error detection.

Here, we investigate the problem of choosing from the available counter-examples a set, that leads to the smallest possible number of candidate error sites. The decision if a given set of counter-examples is 'good' in this sense is proven to be NP-complete. Therefore, two heuristics will be introduced to choose counter-examples.

Binary Decision Diagrams (BDDs) are used to represent the set of all counter-examples. This allows to evaluate the quality of the heuristics. The first heuristic can be considered as a general guideline, how to choose counter-examples, i.e. it gives an idea how to produce good counter-examples by other approaches than BDDs, e.g. by simulation based methods. The second heuristic efficiently chooses counter-examples from BDDs.

2. Preliminaries

In formal verification commonly sequential problems are transferred to combinational problems by unrolling the circuit in time or by splitting loops at storing elements as e.g. flip-flops. Due to this combinational circuits are investigated in the following. The library of basic gates contains all Boolean functions.

An erroneous circuit $\mathcal{C}_E$ differs from its specification $\mathcal{C}_S$ due to an error. The functional difference can be observed at an output α upon applying the input assignment a to the primary inputs. The tuple $A = (\alpha, a)$ is called a *counter-example* (to the equivalence of $\mathcal{C}_E$ and $\mathcal{C}_S$).

Let $F_E(\alpha_i)(F_S(\alpha_i))$ denote the function calculated at the primary output α_i of the circuit $\mathcal{C}_E$ ($\mathcal{C}_S$) in terms of primary inputs. Then $F_E(\alpha_i) \oplus F_S(\alpha_i)$ represents the counter-examples observed at α_i. Using a BDD to represent this difference all counter-examples are given.

The erroneous circuit is assumed to have exactly one error of the following types (a subset of the error model in [1]): (1) the function of a node is changed, (2) an extra node is added, (3) a gate is missing at one input to a node, (4) an input is added or (5) an input is removed at a node.

Given a counter-example for an erroneous circuit the candidate error sites are determined by means of the path tracing procedure from [5]. The procedure traces a path from the output where the counter-example is observed back to the primary inputs. At each gate an arbitrary input, that has a controlling value, or all inputs, if none is controlling, are traced backwards[1]. All gates on the sensitized path are candidate error sites.

3. Choosing good counter-examples

Given an erroneous circuit $\mathcal{C}_E$ and its specification $\mathcal{C}_S$ the set of all counter-examples can be determined. The path tracing procedure leads to a set of candidate error sites for each of the counter-examples. The intersection of all these sets gives the minimal set of candidates that can be determined. Usually it is too expensive to use all counter-examples, since the number of counter-examples can be exponential in the number of inputs to the circuits. Therefore, a subset of counter-examples has to be chosen, that leads to a small number of candidate error sites.

Theorem 1. *Given a set of size k of counter-examples for an erroneous circuit. The decision problem, if this set leads*

[1]An input value to a gate is controlling, if it determines the output of the gate regardless of all other inputs values.

Table 1. Results using four counter-examples

Circ.				optimal		random		distance		BDD	
	#In	#Out	#Nodes	avg.	dev.	avg.	dev.	avg.	dev.	avg.	dev.
9sym	9	1	269	1.091	0.234	12.085	20.393	**2.388**	2.323	3.094	6.150
alu2	10	6	261	1.624	0.960	11.513	17.660	**2.519**	2.095	2.988	2.533
alu4	14	8	2416	1.034	0.302	1.101	0.350	1.171	0.542	**1.081**	0.346
apex2	39	3	3227	1.028	0.064	1.661	1.274	**1.083**	0.184	1.406	1.067
apex5	117	88	2734	1.033	0.123	2.201	3.479	**1.171**	0.495	1.422	0.776
cordic	23	2	2103	1.030	0.049	**1.031**	0.051	1.040	0.069	1.036	0.061
dalu	75	16	2699	1.340	0.724	2.329	2.711	**1.785**	2.096	3.719	4.527
e64	65	65	717	1.000	0.000	2.388	2.342	**1.000**	0.000	6.957	12.852
ex4p	128	28	1593	1.157	0.625	1.988	4.592	**1.257**	0.812	1.626	2.599
ex5p	8	63	1477	1.279	0.791	8.118	23.456	**1.532**	1.147	1.662	1.696
seq	41	35	2991	1.030	0.113	1.429	1.996	**1.227**	0.576	1.333	0.754
t481	16	1	1091	1.011	0.043	1.325	0.573	1.336	0.615	**1.269**	0.488
x3	135	99	1638	1.083	0.208	3.254	4.801	**1.312**	0.582	1.577	0.842

to the smallest number of candidate error sites compared to all other sets of counter-examples is NP-complete.

This theorem is proven by formally defining the problem of choosing counter-examples and establishing a hierarchy of problems that allows reduction to the NP-complete problem *Minimum Cover* [2] in polynomial time. Thus, it is difficult to find the set of counter-examples that leads to the smallest number of candidate error sites - even if all counter-examples are given.

4. Heuristics to choose counter-examples

Choosing the best set of counter-examples, i.e. the set leading to the smallest number of candidate error sites, is difficult. A heuristic to choose a set of k counter-examples from all num counter-examples is needed.

Two heuristics are introduced in this section. The first one can be considered as a general purpose heuristic, that suggests some conditions how to choose counter-examples. The second heuristic aims at efficiently choosing counter-examples, once their representation by a BDD is given.

The intuition behind both heuristics is to choose a set of counter-examples that are as different as possible.

4.1. Maximum Distance Heuristic

This heuristic makes use of a distance defined between two counter-examples, based upon three observations:

1. Non-specified inputs in a counter-example lead to non-defined outputs that are not marked.
2. Different input-values lead to different assignments of controlling values, i.e. different sensitized paths.
3. Observing errors at different outputs, leads to different sensitized paths.

A greedy-algorithm chooses counter-examples such that the sum of pairwise distances is maximized. The greedy-algorithm runs in time $\mathcal{O}(num \cdot k)$.

4.2. Efficient heuristic on BDDs

Having all counter-examples given by BDDs a more efficient heuristic is desirable. The algorithm explained in this section runs in $\mathcal{O}(k \cdot n)$, where n is the number of inputs.

To get counter-examples from different outputs, the same number of counter-examples is taken from the different BDDs representing counter-examples at different outputs. Using a flag, which BDD-nodes have already been visited, assures that no identical counter-examples are chosen. Preferring short paths leads to a large number of don't cares.

5. Experimental Results

Benchmarks were taken from the LGSynth93 set. Into each of the circuits randomly an error was injected. Only those erroneous circuits were considered where the number of counter-examples was more than 50 (to have a search problem) and less than 140 (to be able to determine the optimal choice). For each circuit 100 erroneous instances were generated and diagnosed by 5 techniques: (1) All counter-examples were taken into account; (2) the optimal choice of 4 counter-examples was calculated; counter-examples were chosen (3) randomly, (4) by the maximum distance heuristic and (5) the efficient heuristic for BDDs.

Table 1 gives results for using 4 counter-examples. All results are relative to the optimum of using all counter-examples. Column 'av.' is the arithmetic mean factor of nodes marked by the heuristic over nodes marked by all counter-examples. Column 'dev.' is the standard deviation.

The heuristics yield better results than random choice and are often almost optimal. The heuristic for BDDs has drawbacks on some circuits compared to the distance heuristic.

References

[1] M. S. Abadir, J. Ferguson, and T. Kirkland. Logic verification via test generation. *IEEE Trans. on CAD*, 7:172–177, 1988.

[2] M. Garey and D. Johnson. *Computers and Intractability - A Guide to NP-Completeness*. Freeman, San Francisco, 1979.

[3] V. Paruthi and A. Kuehlmann. Equivalence checking combining a structural SAT-solver, BDDs, and simulation. In *Int'l Conf. on Comp. Design*, pages 459–464, 2000.

[4] A. Veneris and I. N. Hajj. Design error diagnosis and correction via test vector simulation. *IEEE Trans. on CAD*, 18(12):1803–1816, 1999.

[5] S. Venkataraman and W. K. Fuchs. A decuctive technique for diagnosis of bridging faults. In *Int'l Conf. on CAD*, pages 562–567, 1997.