# Approximate Equivalence Verification of Sequential Circuits via Genetic Algorithms

F. Corno, M. Sonza Reorda, G. Squillero

Politecnico di Torino
Dipartimento di Automatica e Informatica
Torino, Italy

http://www.cad.polito.it/

## 1. Introduction

Industrial design flows for electronic circuits typically include at least one optimization step. During this step, the circuit is analyzed and modified in order to improve some specific characteristic, such as speed, size, or power consumption, without modifying its logic behavior. Woefully, exact-by-constructions optimization methods are not always applicable, and designers frequently need to validate the correctness of the optimization process by verifying the equivalence between the optimized design and the original one.

Today, state-of-the-art sequential optimization techniques can handle designs with up to hundreds of flip-flops. However, traditional [2] OBDD-based techniques easily run into memory explosion when trying to verify the equivalence of such designs and many real cases are practically intractable. Thus, designers are interested in new verification methodologies that are not exact, but are always able to produce some result, even though with different levels of confidence.

Recently, a new approximate verification methodology based on Genetic Algorithms (GAs) has been presented [1]. This approach sacrifices verification exactness in order to handle larger designs, and offers designers the opportunity to trade off CPU time with confidence on the result. The experimental results on a prototypical tool called VEGA (*Verification through Genetic Algorithm*) showed that the approximate approach is able to provide results that escaped the reach of an exact verification tool, thus proving that approximate verification is useful when used in conjunction with exact tools.

This paper presents a significant improvement over the VEGA algorithm, allowing us to obtain better results and to deal with larger circuits. In particular, the new algorithm is able to verify circuits with arbitrary optimizations, and not only local modifications. Although the proposed GA is totally different from VEGA, we called it VEGA2 to stress that it shares the same fundamental idea, i.e., approximate equivalence verification through GAs.

## 2. The VEGA2 Algorithm

Given two circuits, a *distinguishing sequence* is an input pattern able to produce an output behavior in one system different from the one produced by the other. If such a sequence exists, an equivalence verification tool should be able to provide it as a *counterexample*; otherwise, it should provide the *proof* of its non-existence. The approximated approach presented in this paper will never be able to provide non-existence (i.e., circuit equivalence) proofs, but is meant to be much more effective in finding the counterexample, when existing. Experimental results will prove that, even with this limitation, the degree of confidence that the approximated result can provide is quite high.

The need for verification can arise during the optimization process, or between two different implementations of the same design. In any case, the circuits have corresponding inputs and outputs, but often the correspondence of many internal points in the circuits is also known, and can be exploited during verification.

In the VEGA [1] algorithm, each gate was assumed to have a corresponding one in the companion circuit, thereby limiting verification to circuits that differ at most in one gate. The improved VEGA2 algorithm presented in this paper, instead, only matches a small set of signal couples that are given by the designer, called *checkpoints* in the following, besides primary inputs and outputs. Depending on the optimization process, it is usually straightforward to identify such corresponding points, where the two circuits are expected to behave equivalently. In the worst case, a black-box verification is performed. Checkpoints are only used as *hints* to the algorithm, and their functional equivalence is not assumed, thereby allowing the user to take benefit even from partial or hypothetical correspondences.

During verification, the circuit is considered as a black-box, where only primary inputs, primary outputs and checkpoints are observable.

The GA, evolving a population of sequences, derives a sequence that causes a difference on the primary outputs, guiding the search by analyzing the differences that may appear on the checkpoints.

Each sequence is characterized by its *fitness*, i.e., its closeness to the goal. The goal is to excite some

differences and to propagate them to a primary output of the circuit. Since circuits are regarded as black-boxes and the location of the differences is unknown, the goal can be rephrased as follows: excite all possible behaviors in the circuit while trying to retain any difference found in some checkpoints. The following fitness function $F(s)$ for a sequence $s$ is thus used in VEGA2:

$$F(s) = \sum_{vectors \in s} (\alpha_1 C + \alpha_2 C_{retain} + \alpha_3 A)$$

where, for each vector in $s$, $A$ estimates the circuit activity, i.e. the number of non-repeated gate and flip-flop value inversions, $C$ counts the number of checkpoints with different values in the two circuits, and $C_{retain}$ the number of such checkpoints whose difference does not disappear in the following clock cycle. The coefficients $\alpha_1 > \alpha_2 > \alpha_3$ set the relative importance of the sub-goals, in increasing order: letting the circuit explore new configurations, retaining a difference on a checkpoint, forcing a new difference on a checkpoint.

## 3. Experimental Evaluation

A prototypical implementation of VEGA2 has been developed using the ANSI-C language and tested on most ISCAS89 benchmark circuits. VEGA2 includes an in-house developed, 3-valued, event-driven, gate-level simulator that is used to compute the fitness function values. All experiments were performed on a Sun SPARCstation 5/110. All programs were limited to use no more than one hour CPU time per circuit and 70 Mbytes of memory.

The experiments mimic the subtle errors that can occur in an optimization process, that the verifier is aimed at discovering. The supposed error model is that of a slight modification in the combinational part of the circuit (modification of sequential elements would be easier to detect, since they have wider effects). Please note that different combinational parts do not imply sequential non-equivalence, since reachable states must be taken into account, and a real sequential equivalence algorithm must be used.

Two error injection models have been considered here: the first one, called *Single Gate Model* (SGM) is the same adopted in [1]. The optimization process is simulated by modifying the type of a single combinational gate, for instance, transforming a NOR gate in an AND gate. In this model, the circuits are very likely to be different, and unrealistic conclusions might be drawn.

The second error injection model is more accurate and it is called *Fanout-Free Region Model* (FFRM): a fanout-free region (FFR) is randomly selected, a single bit in its truth table is flipped and the updated FFR is synthesized. Since no checks are performed to determine if the input configuration for the flipped bit is sequentially reachable or to determine if the flipped bit is sequentially observable, circuits are often sequentially equivalent.

Using the SGM all experiments reported in [1] have been redone. In the 27.06% of the experiments VEGA2 found a distinguishing sequence while AQUILA did not provide any answer. On the contrary, the OBDD-based tool disproved equivalency in the 0.59% of the cases where VEGA2 did not find any distinguishing sequence. But, these results are not surprising and they confirm the superiority of the GA reported in [1]. Moreover, VEGA2 is on the average 10 times faster (64,038 seconds per run against 6,478 seconds) and require less memory.

480 different tests were run on the ISCAS89 benchmarks adopting the FFRM. Results show that for the 27.08% of the circuits, VEGA2 is able to disprove an equivalency while the OBDD-based tools cannot provide any results. On the other hand, in only the 0.42% of the cases AQUILA disproved equivalency while VEGA2 failed to find a valid distinguishing sequence. Moreover, also in this test VEGA2 is on the average 10 times faster than AQUILA (17,642 seconds against 1,762).

## 4. Conclusions

We presented VEGA2: a Genetic Algorithm-based approach to the problem of equivalence verification. Although sacrificing the exactness of the verification, the advantages of such an approach lie in the ability to handle large designs and in the possibility to easily trade off CPU time with confidence on the result (by tuning the maximum number of generations).

VEGA2 is not a replacement for exact verification tools, but a complement: when the complexity of the circuits prevents the use of a BDD-based algorithm, it is still able to provide meaningful results.

We also presented a prototypical tool and experimental analysis that shows that VEGA2 is able to provide a larger number of correct results than both an exact method and the previous GA-based approach. Thus it is able increase confidence on the validity of an optimization process.

Moreover, compared with VEGA, it has an higher degree of confidence (experimental results show that in 9.06% of the tests VEGA2 disprove the equivalency while VEGA did not, and the contrary happened in the 0.29% of the tests) with significantly less restrictive hypothesis on the circuits and with less information on the internal behaviors of the systems under exam.

## 5. References


[1] F. Corno, M. Sonza Reorda, G. Squillero, "VEGA: A Verification Tool Based on Genetic Algorithms," *International Conference on Circuit Design*, 1998

[2] A. Ghosh, S. Devadas and A. R. Newton, *Sequential Logic Testing and Verification*, Kluwer Academic Publishers, 1992

[3] S.-Y. Huang, K.-T. Cheng and K.-C. Chen, "AQUILA: An Equivalence Verifier for Large Sequential Circuits," *ASP-DAC*, 1997