Theoretical fundamentals of functional verification based on random test benches

I. Ugarte, P.Sanchez
Microelectronics Engineering Group. TEISA Department. ETSIIT. University of Cantabria
Avda. los Castros s/n. 39005 Santander. Cantabria. Spain
{ ugarte, sanchez }@teisa.unican.es

Abstract –

The latest releases of commercial verification tools often include coverage metrics and constraint-based generation of random test benches. This commercial interest may be provoked by several factors. Firstly, it seems that currently, the best method of functional verification automation is the coverage-driven random-based test bench generation. Secondly, designers show confidence in the results of random-based functional verification and there is even the impression that these test benches work better for verification than for test. This impression is mainly supported by practical experiences because although there is a huge amount of work on the theoretical bases of random testing, as far as we know, there is no similar work on functional verification.
The main goal of this paper is to explore theories that can analyze random-based functional verification methodologies. They are based on polynomial models of the system under verification and they provide fault-model independent coverage.

I. INTRODUCTION

The latest version of the International Technology Roadmap for Semiconductors, ITRS [1], highlights that verification has become the dominant cost in the design process. Amongst the short-term verification challenges, the document includes the need to quantify the quality of the verification effort, in particular, the need to provide a meaningful notion of coverage.
Traditionally, the quality of a set of test benches is evaluated with coverage metrics [2], such as code coverage (statement/block and path coverage) and control coverage (for example, branch or transition coverage). In this paper, only code coverage metrics will be considered. The goal of statement coverage is to identify sections of code that are not executed. Path coverage provides the percentage of execution paths that has been activated.
The latest releases of commercial hardware simulators normally support these metrics along with constraint-based random test bench generators and verification-oriented languages (PSL/Sugar, OpenVera, etc).
Other approaches evaluate the quality of the verification test benches with behavioral fault models based on the classical stuck-at fault model [3] or they use observability-oriented techniques [4].
This paper introduces a methodology that allows estimating the fault coverage of a set of random test benches when all the possible faults are considered. This magnitude is used to provide a meaning to path coverage metric. The analysis is based on a polynomial model of the system under verification and makes use of numerical analysis and number theory results.

II. SYSTEM MODEL

In this paper, we assume that the system is described at behavioral level as a set of statements that operate with integer data. Some basic operators are supported (addition, subtraction, multiplication, relational and logic operators) as well as 'if' control statements.
Every execution path in the behavioral description can be modeled with polynomials (which model the system behavior) and a set of constraints, which model the if-statement conditions.

III. THEORETICAL FUNDAMENTALS

The goal of this paper is not to develop new mathematical theories for functional verification. Instead of this, some theories from computer algebra and number theory will be used to understand some verification results.
Let r be the number of variables or dimensions of one of the polynomials that model the system. Let $X=(x_1,\ldots,x_r)$ be any point of the input space. Two types of input spaces or domains are considered: real (R^r) and q-element discrete (F_q^r) input domains (where q is a prime power, $q=k^m$, with k prime and m a positive integer).
The "Lagrange Interpolation Problem" [5] can be stated in the real domain in the following form: Given a finite number of R^r points ($X_1,\ldots, X_N$) and some real constraints $y_1,\ldots,y_N$, find a polynomial $p\in R[x_1,\ldots,x_r]_d$ (subspace of r-variable polynomials of total degree of at most d), such that:

$$p(X_j)=y_j \qquad j=1,\ldots,N$$

In particular, we are interested in values of N for which there exists only one polynomial p. From the functional verification point of view, this set of N inputs is the maximum number of test benches that have to be applied to a polynomial model in order to guarantee that there is only one possible behavior.

Counting zeros of polynomials in finite fields is an important topic in number theory. In [6], the number of polynomials that has zeros in a set of N points of F_q^r is specified in Lemma 4.1:

LEMMA 4.1 [6]: Let A be a subset of F_q^r of size N such as that N.C(μ) <d. The number of polynomials of F_q [x_1,…,x_r]$_d$ having a zero of multiplicity of at least μ at all the points of A is

$$\left(\frac{1}{q^{C(\mu)}}\right)^N \times \#\mathrm{F_q}\,[\mathrm{x}_1,\ldots,\mathrm{x}_r]_\mathrm{d}$$

From the verification point of view, the consequence of this lemma is that, if we consider all possible bugs/faults, the probability that a set of N random test benches will not detect a bug/fault will be:

$$\mathrm{P_{No\_detect}} = \left(\frac{1}{q}\right)^N \qquad (1)$$

Where q is the number of different values that any input can take, and N the number of applied test benches.

IV. Functional Verification Of Data Statements

A consequence of the theory in section 3 is that the number of exhaustive test benches is independent of the input ranges (or number of bits). Thus,

<u>Proposition 1:</u> *The number of test benches that completely verify a model depends on the maximum polynomial degree and number of inputs, but it is independent of the input range.*

The consequences of this result are important. Several verification techniques propose transforming the integer inputs into bit vector inputs. This decomposition transforms an integer model into a binary model but it increases the number of variables and the total degree of the polynomial. Thus the verification complexity has been exponentially increased with this transformation.
A direct conclusion of equation (1) is that after N test benches are applied, the coverage of all possible faults will be $(1-\left(1/q\right)^N)$, where q is the number of possible values of the variables.
Thus, considering the metrics that can identify all the behavioral polynomials (path coverage), we conclude:

<u>Proposition 2:</u> *When a path is executed, the probability that a faulty behavior is detected in an output is $1-\left(1/q\right)$, where q is the number of input values.*

Finally, it can be directly concluded that, the product of the percentage of covered path (verified polynomials) and the probability in Proposition 2 provides an estimation of the fault coverage of the data statements. Thus,

<u>Proposition 3:</u> *The coverage of all possible data-statement faults can be estimated by:*

$$\mathit{Fault\_coverage} = \mathit{path\_coverage} * (1-\left(\frac{1}{q}\right))$$

V. Conclusions

This paper presents a methodology for analyzing the coverage of functional verification metrics. This polynomial-based methodology has been applied to descriptions that only include data statements.
The main conclusion of the "path coverage metric" analysis is that it will provide a high coverage if the input range is big enough. The work also defines an equation that allows estimating the fault coverage of a sequence of N test benches. This equation is used to estimate the total fault coverage that a path coverage value provides.
Another conclusion of this work is the evaluation of the exhaustive system verification complexity. This complexity depends on polynomial degrees and input number, thus transformations that increase these values (for example, integer to bit vector conversion) can have a strong impact on the verification complexity.

VI. References

[1] International Roadmap for Semiconductors.2003 Edition, in http://public.itrs.net/

[2] J. Bergeron, "Writing Test benches: Functional Verification of HDL Models. 2nd Edition. Kluwer Academic Press. 2003.

[3] F. Ferrandi, F. Fummi, D. Sciuto, "Test Generation and testability Alternatives exploration of critical algoritms for embedded application". IEEE Trans. On Computer. Vol 51, no 2, February 2002.

[4] J. Costa, S. Devadas, J. Monteiro,"Observability Análisis of Embedded Software for Coverage-Directed Validation". Proceeding of ICCAD'00. 2000.

[5] M. Gasca, T. Sauer, "Polynomial interpolation in several variables". Advances in Computational Mathematics, Vol 12, 2000.

[6] J. Ragot, "Counting Polynomials with Zeros of Given Multiplicities in Finite Fields". Finite Fields and Their Applications, Vol 5, 1999.