Fault Detection of System Level SoC Model

Vladimir Hahanov, Eugenia Litvinova, Yulia Hahanova, Wajeb Gharibi

Abstract – **The effective process models and methods for diagnosing the functional failures in software and/or hardware are offered. The register or matrix (tabular) data structures, focused to parallel execution of logic operations, are used for detecting the faulty components.**

Keywords – **testing, verification, HDL-model, Infrastructure IP.**

1. INTRODUCTION

The objective of the research is to reduce time-to-market and improve the quality of digital systems-on-chips by developing the assertion-based infrastructure, models and methods for verification and diagnosis HDL-code. The information, needed for detecting failures at the functional blocks, is formed during simulation (execution) of software code [1-2]. Design effectiveness for digital product is determined as the average and normalized in the range [0,1] integral criterion:

$$E = F(L,T,H) = \min[\frac{1}{3}(L+T+H)],\ Y = (1-P)^n;$$

$$L = 1 - Y^{(1-k)} = 1-(1-P)^{n(1-k)};$$

$$T = [(1-k)\times H^s]/(H^s + H^a);\ H = H^a/(H^s + H^a).$$

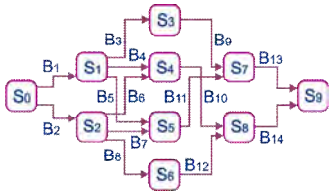
The criterion takes into account the following: the error level L, the verification time T, software-hardware redundancy, determined by the assertion engine and Infrastructure IP tools H. The parameter L, as a complement of the parameter Y (yield), depends on the testability k of a design, the probability P of existence of faulty components, and the quantity of undetected errors n. The time of verification is determined by the testability of a design k, multiplied by the structural complexity of hardware-software functionality, divided by the total complexity of a design in code lines. The software-hardware redundancy depends on the complexity of assertion code and other costs, divided by the total design complexity. At that software or hardware redundancy has to provide the specified diagnosis depth for functional errors and time-to-market, defined by customer.

II. MODEL FOR DETECTING FUNCTIONAL FAILURES IN SOFTWARE

An analytic model for verification of HDL-code by using temporal assertion engine (additional observation lines) is focused to achievement the specified diagnosis depth and presented as follows:

$$M = f(F, A, B, S, T, L), \quad F = (A*B)\times S;\ S = f(T, B);$$
$$A = \{A_1, A_2, ..., A_i, ..., A_n\};\ B = \{B_1, B_2, ..., B_i, ..., B_n\};$$
$$S = \{S_1, S_2, ..., S_i, ..., S_m\};\ S_i = \{S_{i1}, S_{i2}, ..., S_{ij}, ..., S_{ip}\};$$
$$T = \{T_1, T_2, ..., T_i, ..., T_k\};\ L = \{L_1, L_2, ..., L_i, ..., L_n\}.$$

Here $F = (A*B)\times S$ is functionality, represented by Code-Flow Transaction Graph – CFTG (Fig. 1); $S = \{S_1, S_2, ..., S_i, ..., S_m\}$ are nodes or states of software when simulating test segments. Otherwise the graph can be considered as ABC-graph – Assertion Based Coverage Graph. Each state $S_i = \{S_{i1}, S_{i2}, ..., S_{ij}, ..., S_{ip}\}$ is determined by the values of design essential variables (Boolean, register variables, memory). The oriented graph arcs are represented by a set of software blocks $B = (B_1, B_2, ..., B_i, ..., B_n)$, $\bigcup_{i=1}^{n} B_i = B$; $\bigcap_{i=1}^{n} B_i = \varnothing$, where the assertion $A_i \in A = \{A_1, A_2, ..., A_i, ..., A_n\}$ can be put in correspondence to each of them. Each arc B_i – a sequence of code statements – determines the state of the node $S_i = f(T, B_i)$ depending on the test $T = \{T_1, T_2, ..., T_i, ..., T_k\}$. The assertion monitor, uniting the assertions of node incoming arcs $A(S_i) = A_{i1} \vee A_{i2} \vee ... \vee A_{ij} \vee ... \vee A_{in}$ can be put in correspondence to each node. A node can have more than one incoming (outcoming) arc. A set of functionally faulty blocks is represented by the list $L = \{L_1, L_2, ..., L_i, ..., L_n\}$. The model for HDL-code, represented in the form of ABC-graph, describes not only software structure, but test slices of the functional coverage, generated by using software blocks, incoming to the given node.



$B = (B_1B_3B_9 \vee (B_2B_7 \vee B_1B_5)B_{11})B_{13} \vee$
$\vee ((B_1B_4 \vee B_2B_6)B_{10} \vee B_2B_8B_{12})B_{14} =$
$= B_1B_3B_9B_{13} \vee B_2B_7B_{11}B_{13} \vee B_1B_5B_{11}B_{13} \vee$
$\vee B_1B_4B_{10}B_{14} \vee B_2B_6B_{10}B_{14} \vee B_2B_8B_{12}B_{14}.$

Fig. 1. ABC-graph for HDL-code

III. CONCLUSION

A new model of software in the form of Code-Flow Transaction Graph, as well as a new matrix method for diagnosing functional failures, which are characterized by adaptability of data preparation when detecting faulty blocks, are proposed. They allow considerably reducing the design time of digital systems on chips.

REFERENCES

[1] V.I. Hahanov, I.V. Hahanova, E.I. Litvinova, O.A. Guz, "Design and Verification of digital systems on chips," *Kharkov: Novoye Slovo*, 528 p., 2010.

[2] E.J. Marinissen, Yervant Zorian, "Guest Editors' Introduction: The Status of IEEE Std 1500," *IEEE Design & Test of Computers*, No26(1), pp.6-7, 2009.