Transaction Level Fault Simulation, Recovery, and Mapping Tool

Fatemeh Javaheri, Zainalabedin Navabi
University of Tehran,
College of Engineering,
School of electrical and Computer Engineering
{n-javaheri, navabi}@cad.ut.ac.ir

***Abstract*— We are presenting a package for hierarchical testing and debugging of systems. It provides testing capabilities at the design specification level before the actual design of the hardware to eliminate many redesign iterations that are necessary for making circuits testable. This package consists of several built-in methods for fault recovery. It includes a library of Timed Automata model of communications in various abstraction levels to perform the fault simulation and localization automatically, facilitating multi-level debugging, and mapping TL faults to stuck-at gate level faults.**

Keywords-TLM; formal verification; design error; timed automata; mutated model

I. Introduction

Over the last fifty years, advances in semiconductor technology, the increasing complexity of digital systems and decreased time to market have raised level of design from transistor to system level. Along this path, design tools have been developed that aid designers in thinking of a design at higher level of abstraction. On the other hand, digital system *testing* has not benefited as much as digital system *design* from system level methodologies, tools, and utilities. To cover this gap, new design and test methods and using of high level languages for test purposes should be considered.

This work is a fully-automated tool set that facilitates injection of high level faults in pre-designed specifications of system level communication links. The environment we are providing is a co-simulation platform for mixed-level evaluation of faulty and good circuit behaviors. It is capable of debugging communications and recovering from a fault.

II. The Package Environment

This package includes several libraries and utilities for performing high-level fault simulation, and mapping transaction level faults to stuck-at gate level faults. Moreover, it is able to localize a fault, and hence, makes it possible to recover from the fault. Figure 1 represents the package environment.

For automating and facilitating the process of test, recovery, and fault mapping, libraries of the Timed Automata [1] representation of the TL communication scenarios [2] and standard on-chip communications are included in this package. Communication scenarios are TLM correspondence of actual handshaking and data handling events that can happen in RT level circuit. Moreover, the package contains a high-level fault model definition for abstract communications. It also contains signatures that are defined based on the abstract communication behaviors. Below, the utilities of the package are discussed.

A. Transaction Level Fault Simulation

For performing the fault simulation, the tool gets the golden model's TA representation of a CS and the attributes that introduce the terms that can be affected by the proposed fault model. As shown in Figure 1, TA model of the faulty communication links is generated using the input TA model by considering the defined faults and attributes. For detecting the effect of a fault, our environment simulates the mutated models by calling the UPPAAL verifier. For each fault, the defined signatures are verified for a given set of input data. The fault is detected if there is at least one property has not been satisfied.

B. Fault Mapping

The package can determine the high-level faults that can correspond to gate-level stuck-at faults, by considering the TA model of TL and standard on-chip communications. For each standard on-chip communication that corresponds to an abstract communication, the mutated models are generated considering the stuck-at faults. Then, the mutated TA of an on-chip and TL communication are considered, and the faults occurred in the equivalent parts of them are categorized to a group.

C. Fault Recovery

Based on the property that is not satisfied in a mutated model, the location of the fault is determined. If it is possible, the communication is recovered from a fault by substituting the faulty path with a possible fault free one.

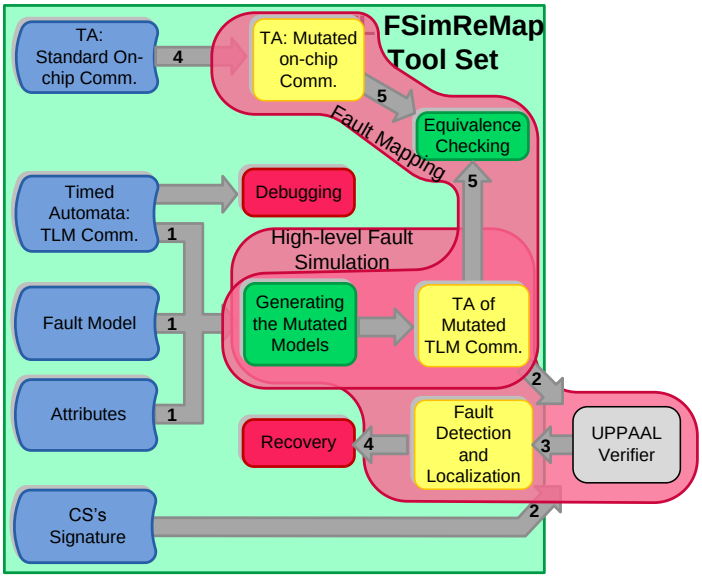

Figure 1. Package environment: performing test, fault recovery, and mapping

III. References

[1] R. Alur، C. Courcoubetis، D.L. Dill، "Model checking for real-time systems"، *In Proc. Symposium on Logic in Computer Science*، 1990، pp. 414–425

[2] F. Javaheri, M. Namaki, P. Kamranfar, Z. Navabi, "Mapping transaction level faults to stuck-at faults in communication hardware".In *Proc. Of 20th Asian Test Symposium 2011 (ATS'11)*