Tallinn University of Technology

Department of Informatics

Chair of Software Engineering

Agnent based hardware design testing framework

Author: Jaak Kõusaar

Supervisor: Kuldar Taveter

Tallinn 2014

# Introduction

This project presents an agent based framework that can be used to test system-on-chip (SOC) design’s durability to arbitrary faults, inserted into the design in a statistical manner. The SOC under test is considered to have a built in self-test (BIST) capability. This project does not include a human factor, all the artificial faults are created randomly, taken into account a predefined variable, which is a combination of faults density and their severity level. The decision whether SOC under test passes the test, is made upon the SOC’s ability to carry out its tasks in a certain period of time, despite the erroneous system. These tasks always require an involvement of more than one system component.

# Motivational layer

## Goal model

Goal model of hardware design testing framework is given in figure 1. The main goal of described system, is to keep always performing defined tasks. The tasks are defined by the design under test that represents a piece of hardware that is composed of partially autonomous nodes. To keep the performing tasks, system has to test itself starting from the startup moment and carrying on testing activities periodically on every node. Every node has to project its tests, based on previous state. This goal is assigned to role tester. After tests are finished, given results must be analyzed and interpreted by the role reporter, whose goal is to properly present test results as structured information. These test results are shared between nodes when asked or forwarded to neighboring nodes whenever new test results are available. This is a task of a role information exchanger. Systems main goal, performing defined tasks, can be divided to separate processes that in turn can be mapped to different nodes. This mapping is done by the role planner, based on available information on the nodes states whenever an input signals are given.



Figure 1. Goal model

## Role models

5 roles presented in the system, are in more detail described below

|  |  |
| --- | --- |
| Role name | System |
| Description | Main role - a core of framework |
| Responsibilities | Carry out hardware design tasks i.e. take inputs and produce output |
| Constraints | Must always perform defined tasks  Must periodically perform tests  Processes must be performed regardless of current state |

|  |  |
| --- | --- |
| Role name | Tester |
| Description | Component that creates test packets for the node it represents |
| Responsibilities | Test projection |
| Constraints | Projected tests must be based on previous test results |

|  |  |
| --- | --- |
| Role name | Reporter |
| Description | Component that takes test results and interprets them in usable way |
| Responsibilities | Present test results |
| Constraints | Presented test results must be analyzed and composed in structured way |

|  |  |
| --- | --- |
| Role name | Planner |
| Description | Component that takes well-formed test results and makes process mapping for the node it represents, based on those results |
| Responsibilities | Map processes |
| Constraints | Mapping is done only when according input is given  Mapping task is done, based on the test information of that node |

|  |  |
| --- | --- |
| Role name | Information exchanger |
| Description | Gives information about the health state of the node it represents or passes information of acquired from its neighboring nodes |
| Responsibilities | Transmit information |
| Constraints | Information is given if it is asked  Information is given to neighboring nodes, whenever new test information is available |

## Organization model

Organizational model, showing relationships between the roles, is presented in figure 2. System controls Tester, giving to Tester an order to test the node. System also controls Planner, forwarding processes as an input. Tester interacts with Reporter, passing last test information. Reporter on the other hand, can provide Tester with earlier test results as basis for next tests. Reporter can also initiate testing process, by passing to Tester according request. Reporter interacts with Information exchanger, by providing latter with analyzed and properly structured test results. Information exchanger can also request these results from Reporter. Planner interacts with Information exchanger, asking for available data on host node or its neighboring nodes. Planner can also ask Information exchanger to pass data for processing to neighboring nodes.



## Domain model

Resources are divided into following entities:

* List of all nodes
* List of neighboring nodes
* Task
* Test results
* Test

