Abstract

This paper describes the results of the COPERNICUS europroject JEP-97-7133 VILAB (Virtual LABoratory) on creating an environment for internet-based collaboration in the field of design and test of digital systems. Different CAD tools at geographically different places running under the virtual environment using the MOSCITO system can be used for microelectronics design, fault simulation and test generation purposes. The interfaces between the integrated tools were developed during the project work. The tools can be used separately, or in multiple applications in different complex flows. The functionality of the virtual laboratory in a collaborative experiment on defect-oriented fault simulation and test generation was tested and is described in the paper.

1. Introduction

Three years ago a new joint europroject JEP-97-7133 VILAB in the frame of COPERNICUS was started with the goal to create a virtual laboratory for cooperative research in the field of dependable microelectronics system design.

In the field of digital design, system-on-chip (SoC) technology is becoming state-of-the-art. The design of SoCs raises a lot of EDA problems: HW/SW codesign, high-level synthesis, testability evaluation, test pattern generation, fault simulation, physical defect analysis with respect to the whole system to be integrated. Usually not all the needed EDA tools are available for a designer in his working site. Now, the Internet opens a new dimension, and offers new chances using tools from different places.

2. MOSCITO

The MOSCITO system offers a Client-Server concept. There is one Master Server, several Slave servers and an arbitrary number of clients. The requested service is provided by Slave servers. That is because so-called Agents were attached to each Slave server. These Agents encapsulate service providing work tools (program executables). An Agent can be seen as an intelligent wrapper around another stand-alone program, and it is capable to communicate with Servers. At any time, it is possible to add and remove Agents, respectively. All Slave servers are registered at the Master Server, so all Agents (i.e. available services) are also registered at the Master server. Users access first the Master server and will get a list of services provided. After selecting a service (Agent) wanted, the user is automatically re-
directed to Slave server, and after that the work with the service providing tool can start.

The main emphasis in the tool integration was put on the following aspects:

- Encapsulation of design tools and adaptation of the tool-specific control and data input/output to the MOSCITO framework.
- Communication between the tools for data exchange to support distributed Internet-based work.
- Uniform graphical user front-end program for the configuration of the tools, the control of the whole workflow and the visualization of result data.

Moreover, an important goal is to provide the functionality of a tool to a potential user as a service in a local area network (LAN). This approach is similar to the Application Service Provider (ASP) idea or the recent approach of Web Services.

In the present system as the result of the project VILAB, the following tools have been integrated in MOSCITO (Fig.1):

- Interface to system-level HW/SW co-design environment (C²VHDL code migration tool) [3].
- Behavioral ATPG [4].
- High-level synthesis system CAMAD [5].
- Interface from RTL VHDL to hierarchical ATPG.
- Interfaces from low-level EDIF and ISCAS formats to hierarchical and low-level ATPGs and fault simulators.
- Hierarchical ATPG DECIDER [6].
- Logic level fault simulation and test generation tools Turbo-Tester [7].
- DefGen - ATPG for $I_{DDQ}$ and voltage testing of digital circuits [8,9].
- Tst2Alb - a data converter between ATPG tools
- ALB - an automatic fault library builder [10].

The tools can act as MOSCITO agents and each of them provides a demanded service. The user is empowered to combine all the services to a problem-specific workflow. That means, the needed tools have not to be installed on the users local computer. The user’s effort for installation, configuration and maintenance of software will be drastically reduced. Furthermore, specialized tools can be executed on their native platform with a high performance (e.g. supercomputer with fast CPUs and large memory, Workstation-Cluster). To facilitate remote computing in this way is important for application with huge amount of computing time: e.g. fault simulation as well as test pattern generation.

The MOSCITO framework is implemented in JAVA, and can run on different computing platforms. The only prerequisite is an installed Java Virtual Machine. At the moment MOSCITO is used on SUN workstation (Solaris) and on PCs (Microsoft Windows and LINUX). The detailed description of the VILAB environment - software architecture of MOSCITO, tool encapsulation methodology, communication methods, graphical user interface and Internet-based usage is given in [2].

To validate the MOSCITO system and to collect experience while using it for real-life applications, an experimental workflow for defect-oriented test pattern generation was developed and mapped to a MOSCITO workflow. In the following the functionality of the tools used in this workflow is explained.

![Fig. 1. Integrated tool environment](image-url)
3. Tool descriptions

3.1. Fault simulation and test generation

The Turbo Tester ATPG software (Block 8 in Fig.1) consists of a set of tools (Fig.2) for solving different test related tasks by different methods and algorithms:
- test pattern generation by deterministic, random and genetic algorithms;
- test optimization (test compaction);
- fault simulation and fault grading for combinational and sequential circuits;
- defect-oriented fault simulation and test generation;
- multi-valued simulation for detecting hazards and analyzing dynamic behaviour of circuits;
- testability analysis and fault diagnosis.

The tools run on the structural logic level. Two possibilities are available - gate-level and macro-level. Using the macro-level helps to reduce the complexity of the model and to improve the performance of tools. The fault model in Turbo Tester is the traditional stuck-at model. However, the fault simulator and test generator can be run also in the defect-oriented mode, where defects in the library components can be taken into account.

3.2. Defect-oriented ATPG

The DefGen ATPG system (Block 12 in Fig.1) is a hierarchical ATPG for combinational circuits for Ismo and/or voltage testing. The random, deterministic TPG algorithms and a fault simulator are involved in the system. The ATPG uses functional fault model and runs over the functional test set specified for each functional cell of a circuit. The functional test set for each cell is a part of the fault conditions library for DefGen. These lists can be created e.g. from a defect analysis of circuits cell at the low level or can be specified by the designer with regards to the used fault model for the investigated cells. The input format for circuit description is the language from ISCAS’85 benchmark circuits. The EDIF-ISCAS translator from Turbo Tester is used as the interface to DefGen.

An Automatic Fault Library Builder (ALB) has been developed and implemented for finding an optimal functional patterns for cells in the circuit [10].

TPG technique at higher levels of abstraction rests upon a functional fault model and physical defect - functional fault relationships in the form of a defect coverage tables at the lower level. Each table includes the following information:
- list of possible defects of the cell;
- erroneous functions performed by the faulty cell;
- list of input patterns detecting physical defects;
- probabilities of physical defects (optional).

3.3. Analysis of physical defects in cells

The lists of erroneous functions and local test patterns for detecting physical defects in cells were generated by electrical simulations at the transistor level by WUT [11]. The defect characterization process is computationally expensive, but it is performed only once for every library cell.

The test generation approach is based on the assumption that the majority of defects occur inside the cells and not in the routing between them. Such assumption would not be realistic in the case of older CMOS technologies with two levels of metal and very dense routing. In the state-of-the-art deep submicron technologies still only one or two levels are used inside cells but 6 or more levels of metal are available for routing. More routing levels means lower sensitivity to defects. Routing between the cells is less dense and various nodes are routed at various metal levels. As a result, probability of shorts outside cells is significantly reduced.

Only one type of physical defects in CMOS gates - shorts between conducting regions were considered. This is one of the most important sources of faults in CMOS digital circuits. However, the methodology can be extended to other types of physical defects as well.

Probabilities of defects occurrence were calculated by layout probabilistic analysis at the physical level taking into account defect density and size distribution.

4. Experimental research

The described MOSCITO based environment was tested in the frame of VILAB by the partners IIS/EAS (Germany), IIN (Slovakia), TTU (Estonia) and WUT (Poland). The purpose of experiments was to compare the quality of 100% stuck-at test patterns in relation to physical CMOS defects, and to improve the defect coverage. For that we used the data produced at WUT by probabilistic analysis of physical defects for a restricted library of complex gates. Then we resynthesized the ISCAS’85 circuits using only
components from the analysed library. The problem to investigate was to determine how good are the 100% stuck-at tests in detecting physical defects in complex gates. The results in Table 1 show the low quality of 100% stuck-at tests in detecting real physical defects. Some improvements in quality of tests are shown in the right part of the table were simulation-based defect-oriented ATPG was used. The cooperation between partners is continuing towards developing of deterministic defect-oriented ATPG to achieve further improvements in the test quality.

Authors believe that the MOSCITO architecture is powerful enough to solve similar problems in other application areas of automated system design. Future cooperative work will continue in this direction.

References


<table>
<thead>
<tr>
<th>Circuit</th>
<th>Defect coverage for OR-type shorts, %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Traditional stuck-at test generation</td>
</tr>
<tr>
<td></td>
<td>Counted defects</td>
</tr>
<tr>
<td>C17</td>
<td>92.59</td>
</tr>
<tr>
<td>C432</td>
<td>99.38</td>
</tr>
<tr>
<td>C499</td>
<td>92.80</td>
</tr>
<tr>
<td>C880</td>
<td>95.95</td>
</tr>
<tr>
<td>C1355</td>
<td>93.42</td>
</tr>
<tr>
<td>C1908</td>
<td>92.91</td>
</tr>
<tr>
<td>C3540</td>
<td>94.21</td>
</tr>
<tr>
<td>C5315</td>
<td>94.71</td>
</tr>
<tr>
<td>C6288</td>
<td>92.59</td>
</tr>
</tbody>
</table>

Table 1. Data of defect-oriented fault simulation and test generation

The new result of these experiments was to show that the quality of tests in terms of defect coverage is higher when the defect probabilities are not taken into account. From that we can conclude that the traditional methods of test coverage measuring based on simply counting of not detected defects, where all the faults are assumed to have the same probability, are tending to give overestimated quality measures.

5. Summary

In the paper an Internet-based environment based on MOSCITO system [12] is presented. The environment is focused on providing CAD support in a large field of functionalities: SW/HW cosimulation, high-level and logic level design flows, test pattern generation and fault simulation at behavioral, functional, logic and physical defect levels. The main effort was put on linking together CAD tools and test generators with varying functionalities and diverse fault models available at geographically different sites. The system provides interfaces and links to commercial design environments and also to university CAD tools. The functionality of the environment was verified by several benchmark circuits and by different design and test flows, especially in defect-oriented test generation where some new research results were obtained.