Bilateral German-Estonian project (2004-2006)

Title: Functional Built-in Self-Test in Digital Systems

Principal investigator: prof. R.Ubar

Partners: FhG Institute of Integrated Circuits, Dresden and University Stuttgart

Investigators: E.Ivask, J.Raik

Integration Test Software into e-Learning Environment MOSCITO

Abstract: The project was mainly focussed on a cooperative joint high- and low-level design flow with automated test generation for developing complex digital systems. The substantial R & D goals have been addressed to achieve high testability and high quality testing by efficient test generation and fault simulation.

The features of this project are:

- Internet based partners' competence and tool integration
- contributions to hierarchical tools for test generation and fault simulation.

The most important R & D activities and results were supplied by TTU (Tallinn), IIS/EAS (Dresden) and other partners LIU (Linköping) and TUD (Darmstadt):

- A new hierarchical automated test pattern generator (ATPG) has been developed at TTU for generating deterministic tests for highly sequential complex circuits and systems.
- Two interfaces (VHDL-DD converter and EDIF-SSBDD converter) have been developed at TTU in cooperation with LIU for integrating the hierarchical ATPG and other test oriented tools developed at TTU into commercial design environments (e.g. Synopsys, Mentor Graphic).
- The tools for test generation and fault simulation developed at TTU, and the converters have been integrated into the MOSCITO environment developed at IIS/EAS for Internet based cooperative R&D work. The coordination of these activities and critical analysis of methodologies for Internet based cooperation was carried out by TUD.
- Including new test oriented operations into a commercial standard design environment helps to operatively increase the testability of designs and by that to obtain a higher quality of products.
- A demonstrator has been developed and experimented which includes three different Internet based design-and-test-work-flows as a result of cooperation between four partners where the flows differing in their functionalities can be chosen.
- The scientific mission of joining partners' competence aimed at exploiting synergy resulting in improved design quality and reduced time-to-market has been achieved.

Publications and papers

[1] G.Elst, K-H.Diener, E.Ivask, J.Raik, R.Ubar. FPGA Design Flow with Automated Test Generation. *Proc. of German 11th Workshop on Test Technology and Reliability of Circuits and Systems.* Potsdam, 1999, pp. 120-123. (Joint paper of EAS/IIS and TTU)

[2] K.-H. Diener, G.Elst, E.Ivask, G.Jervan, Z.Peng, J.Raik, R.Ubar. Digital Design Flow with Test Activities. VILAB User Forum, Smolenice, April 8, 2000, 11p

[3] K.-H.Diener, G.Elst, E.Gramatova, W.Kuzmicz, Z.Peng, R.Ubar. Virtual Laboratory for Research in Dependable Microelectronics. 7th Baltic Electronics Conference, Tallinn, October 8-11, 2000, pp.217-220. (Joint paper of EAS/IIS, IIN, WUT, LIU and TTU)
[4] A.Schneider, P.Schneider, E.Gramatova, E.Ivask. Internet-basierter Systementwurf mit MOSCITO. In "Entwurf Integrierter Schaltungen", 10. E.I.S. Workshop, Dresden, April 3-5, 2001, pp. 295-296. (Joint paper of EAS/IIS, IIN, and TTU)
[5] E.Ivask, R.Ubar, J.Raik, A.Schneider. Internet Based Test Generation and Fault Simulation. Design and Diagnostics of Electronic Circuits and Systems – DDECS'2001, Györ, Hungary, April 18-20, 2001 (accepted paper).
[6] A.Schneider, E.Ivask, P.Mikloš, J.Raik, K.H.Diener, R.Ubar, T.Cibáková, E.Gramatová. Internet-based Collaborative Test Generation with MOSCITO. IEEE Proc. of Design Automation and Test in Europe – DATE'02. Paris, March 4-8, 2002, pp. 221-226.
[7] A.Schneider, K.-H.Diener, E.Ivask, R.Ubar, E.Gramatova, T.Hollstein, W.Pleskacz, W.Kuzmicz, Z.Peng. Integrated Design and Test Generation Under Internet Based Environment MOSCITO. EUROMICRO Conference, September 3-6, 2002, pp. 187-194.
[8] A.Schneider, K.-H.Diener, E.Ivask, R.Ubar, E.Gramatova, M.Fisherova, W.Pleskacz, W.Kuzmicz. Defect-Oriented Test Generation and Fault Simulation in the Environment of MOSCITO. Proceedings, BEC-2002, Tallinn, October 6-9, 2002, pp.303-306.
[9] A.Schneider, K.-H.Diener, G.Elst, E.Ivask, J.Raik, R.Ubar. Internet-Based Testability-Driven Test Generation in the Virtual Environment MOSCITO. Proc. IFIP Conference on IP Based SOC Design, Grenoble, France, October 30-31, 2002, pp.357-362.
[10] A.Schneider, K.-H.Diener, G.Elst, R.Ubar, E.Ivask, J.Raik. Integration of Digital Test Tools to the Internet-Based Environment MOSCITO. Proc. of 7th World Multiconference on Systemics, Cybernetics and Informatics – SCI 2003. Orlando, USA, July 27-30, 2003, pp.136-141.

Detailed scientific results

Partners' VILAB tool integration based on the MOSCITO system

1 General concept of MOSCITO

Based on the MOSCITO sytem the integration of partners' VILAB tools via Internet is performed. The MOSCITO system, developed and maintained in IIS/EAS Dresden is settled down to an Internet based (TCP/IP sockets) distributed computing conception.
Basically, the MOSCITO system offers a Client-Server concept. There is one Master Server, several Slave servers and 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 (written earlier by third party). An Agent 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.

Features for VILAB tool integration based on the MOSCITO system have been improved considerably in cooperation. So the MOSCITO system has got means for cooperating different CAD tools via Internet. A single control panel (graphical user interface) can be used for tool (services) selection, controlling the execution flow and displaying the (graphical/textual) results. Executing tools via Internet has many advantages. The software that is running in the application server does not have to be installed on client's computer. Client and application software can run on different operating systems. The software can be installed to or accessed from different computers around the net. User benefits immideatly from software tools updates.

2 Tool integration

Several design and test tools can be linked together into one automatic chain-type workflow. At the present, the lenght of the workflow is fixed. Generally, the user selects a workflow size what is suitable for him. An Example flow for test generation tasks is given in figure 1: Workflow of test tasks.
For each tool belonging to the workflow an Java-based wrapper program, the so-called Agent has to be made available. Corresponding links between the modules have to be created using the communication possibilities provided by MOSCITO. This type of

configuration is open and flexible, allowing each Agent to be executed in a different computer.

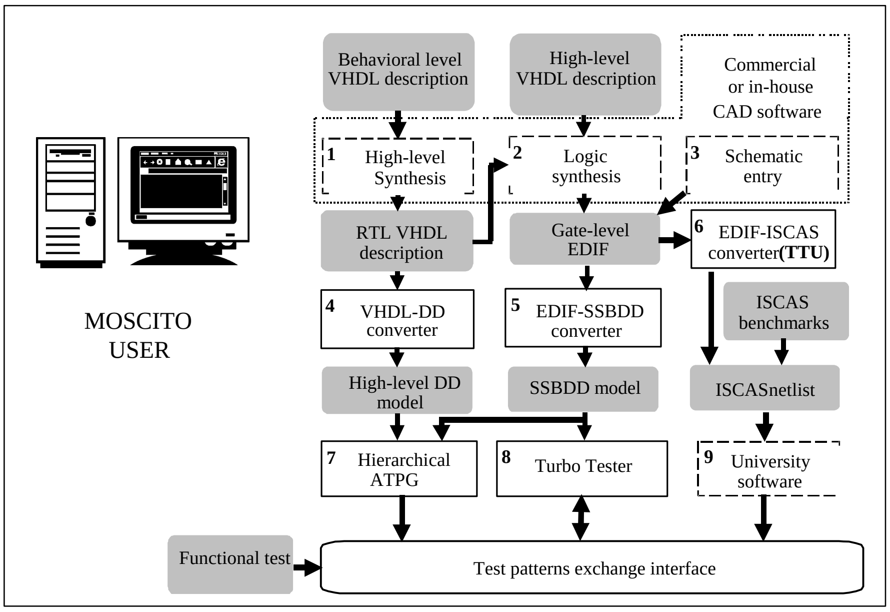

Fig .2 Work flows integrated to the MOSCITO

3 Solving the Firewall tunnelling

At first, MOSCITO was intended for using in a local network and not across firewall protected systems. Therefore, it randomly used the non-restricted communication ports above

1024. It is known that many other network applications also use these so-called free port numbers. There is no harm in internal network generally, but there will be big security problem when such programs are directly exposed to internet. The reason is that some of them are are known to be vulnerable, i.e. they can be misused to attack the host computer they are running on. Tolerating one of such *vulnerable* programs will compromise the host computer and also finally the entire network. Consequently, in a restrictive firewall protected system there are only few ports left open for incoming internet connections (like port 80 for http web

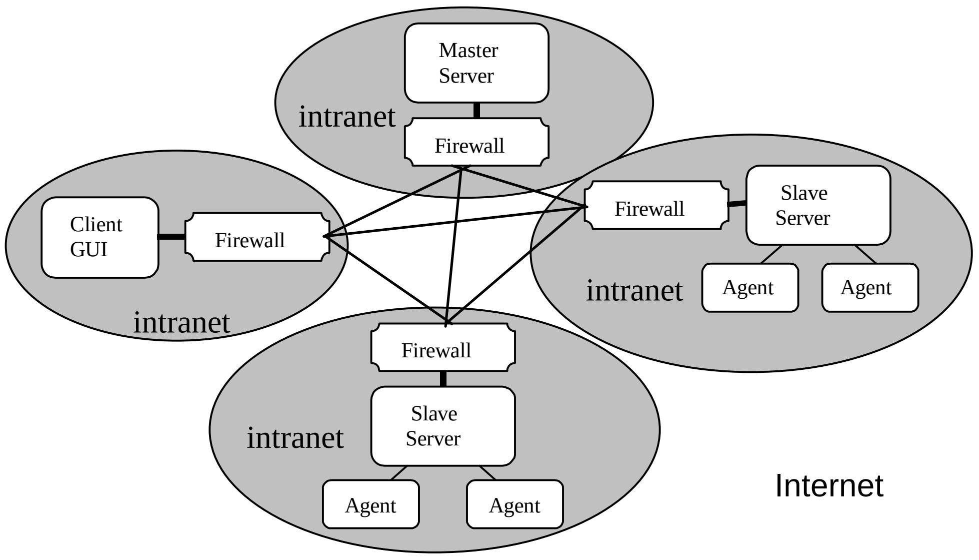

Fig .3 Communication between firewall protected MOSCITO subsystems: connections are allowed only between dedicated communication ports

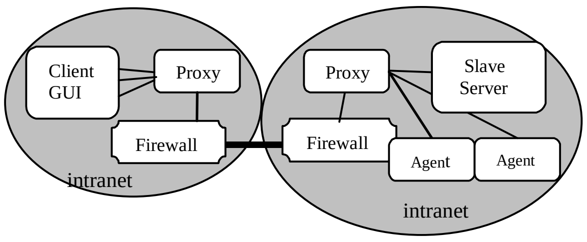

Fig. 4 Communication between client and agents via proxy

server). In the case of restrictive firewall MOSCITO would simply not work, because firewall blocks all the communication.
Previously, the MOSCITO communication strategy required that all the ports above 1024 were configured as open for incoming Internet connections. In order to comply with firewall requirements, the major MOSCITO communication scheme was modified. Now, the MOSCITO communication requires opening only one dedicated port in the firewall. That was

achieved by implementing additional communication layer (so called proxy) between MOSCITO clients and servers. Proxy redirects traffic from several sources to several destinations through single firewall port. Proxy is a piece of software written in Java, reusing partially existing MOSCITO program code.

4 Implementation of the test Agents

There exists straightforward way to integrate a tool into MOSCITO environment. For each Test Tool involved in the work flow there will be one Agent, i.e., for each Test Tool one Java based wrapper is implemented. Control over the tool is performed via the Agent, which executes the program with different parameters.

a) Controlling the Test Flow
There are two levels of automation in controlling the Test Flow:

- Controlling the Test flow is in the user's hands. This is very flexible and convenient for users who prefer having control over the work flow. The tools are selected and executed one after another "manually", not in "batch" mode (automatically).

- Automatic Test flow application. First, the user selects a workflow of appropriate size according to his needs. Then, the user fills the slots of the workflow with tools (Agents). The execution of the workflow is carried out automatically by the MOSCITO core. No any user intervention is needed in between. If necessary, the user can pause, resume or stop the workflow.

b) Communication between tools (Agents)
All test tools read/write go into separate (temporary 'system') folder. During the automatic flow control, the MOSCITO system passes the output of one Agent to another Agent.
While the user is executing single tools he still has the access to results. The user can simply save the result files into folders on his own local computer. If using the automatic flow control, the user can save all the intermediate and final results after the flow is finished.

c) Description of parameters for Test Tools
There is a generic way for how to describe command line parameters for any work tool. Input parameters are described in a XML based configuration file (MML- MOSCITO Markup Language). For each tool (Agent) one MML configuration file exists.

5 Example work flow with MOSCITO

Let us have three remote computers in separate places:

1) One for the MOSCITO Master server (e.g. in Dresden)
2) The second for the MOSCITO Slave server (e.g. in Tallinn)
3) user's computer with MOSCITO Client software (Graphical User Interface (GUI))

We assume that the MOSCITO Master Server is running and on the second computer the MOSCITO Slave server is running. On the second computer we can start for example different Test Agents like the simulation Agent or test vectors generation Agent. As a result, these Agents are automatically registered in the MOSCITO Slave Server. All Slave Servers in turn are registered in the Master Server. Consequently, all Agents can be found via the Master Server. In order to reduce a load on single computer, we can distribute different Test Agents between different computers.

Now, the system is ready to serve the end user on the third remote computer. When the user starts now the MOSCITO Client then it will automatically connect to the MOSCITO Master Server and will receive a list of registered Agents from it. That way, the user can now see and use all the available Test Agents. After a particular testing Agent is selected, a form with required parameters is prompted. These parameters, specified by the user, are forwarded to the MOSCITO Slave server where the parameters are passed to the Test Agent (testing tool). The results of the test tools (e.g. test coverage %, file of test vectors) are first returned to the MOSCITO Slave server which in turn sends them directly back to the user's MOSCITO I/O console.

Now, for example, the user can proceed with selecting another Test Agent depending on his needs, or he can exit the GUI. The Master Server and Slave Server(s) continue their work. For administration purposes, it is possible also remove agents, shut down slave and master servers.

The most important achievments

This project has been addressed to the cooperative development of a novel full design and test generation system with combined high-level synthesis. The most important outcomes are:

- A new hierarchical automated test pattern generator (ATPG) has been developed at TTU for generating deterministic tests for highly sequential complex circuits and systems.
- Two interfaces (VHDL-DD converter and EDIF-SSBDD converter) have been developed at TTU in cooperation with LIU for integrating the hierarchical ATPG and other test oriented tools developed at TTU into commercial design environments (e.g. Synopsys, Mentor Graphic).
- The tools for test generation and fault simulation developed at TTU, and the converters have been integrated into the MOSCITO environment developed at IIS/EAS for Internet based cooperative R&D work. The coordination of these activities and critical analysis of methodologies for Internet based cooperation was carried out by TUD.
- A demonstrator has been developed and experimented which includes three different Internet based design-and-test-work-flows as a result of cooperation between four partners where the flows differing in their functionalities can be chosen depending on the design type and quality demands.

Demonstrator for Design Flow with Automated Test Generation

To demonstrate an Internet-based integrated design and test flow where different tools can run in geographically different places using the MOSCITO virtual environment a Subproject specific demonstrator was defined, agreed and implemented. Now, the necessary implementation activities are going on. Depending on the design problem and designer’s expertise three different design flows may be pursued [6]:

- design with hierarchical and deterministic test generation
- design with low-level simulation-based test generator
- design with user defined functional tests.

The MOSCITO system allows to use the tools involved in the design and test flow over Internet from different geographical places (see Figure 2***:*** Virtual Design and Test Environment)

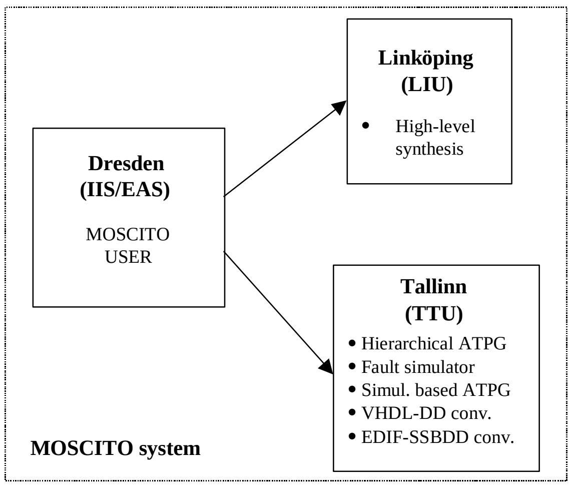

Figure 2: Virtual Design and Test Environment

Test experiments were carried out with the design Rotating detection developed at EAS/IIS. The function of the design is as follows: Given a rotating disc marked with a white and a black sector. The problem is to identify the sense of rotation. Two sensors should make this. A LED for each direction, respectively, should indicate the clockwise rotation and the counter clockwise rotation. If the rotation speed falls below a minimum rev, a LED gives out the signal Stopping. When the rotary speed exceeds a definite rev, one LED for each direction signalises "fast rotation".
If the maximum rev will be exceeded, additionally, a LED for Max displays that case.