High-Level Synthesis and Test in the MOSCITO-Based Virtual Laboratory[1]

A. Schneider, K.-H. Diener
Fraunhofer Institute for Integrated Circuits (IIS/EAS), Germany
{schneider, diener}@eas.iis.fhg.de

J.Raik, R.Ubar
Tallinn Technical University, Estonia
{ieero, raiub}@pld.ttu.ee

G.Jervan, Z. Peng
Linköping University, Sweden
{gerje,zebpe}@ida.liu.se

T. Hollstein, M.Glesner
Technical University of Darmstadt, Germany
{glesner, thomas}@mes.tu-darmstadt.de

Abstract

The paper describes the results of the COPERNICUS europroject JEP-97-7133 VILAB (VIrtual LABoratory) obtained in a Internet-based joint activities of high-level design and hierarchical test generation of digital systems. Different CAD tools at geographically different places running under virtual environment were used for joint research purposes. The interfaces and convertors between the integrated tools were developed during the project. The tools can be used separately over Internet, or in multiple applications in different complex flows. The functionality of the virtual laboratory in a collaborative research on HW/SW codesign, high-level synthesis and test generation was tested and is described in the paper.

1. Introduction

The basic idea of the activities in the project was to exploit Internet-based tool integration. For that purpose several design and ATPG tools implemented at geographically different places were successfully integrated into a new virtual environment MOSCITO [1-3]. The essential features of this integration environment were experimentally proved.

In this paper the results of cooperative research in the field of high-level synthesis and hierarchical test generation between partners from Fraunhofer Institute for Integrated Circuits in Dresden, Technical University of Darmstadt (Germany), Linköping University (Sweden) and Tallinn Technical University (Estonia) are described.

The paper is organized as follows. The MOSCITO system developed in Dresden is shortly described in Section 2. The functionality of the environment via tools description is given in section 3. Experimental results obtained by cooperative use of the environment are shown in section 4.

2. Virtual Laboratory

2.1. MOSCITO

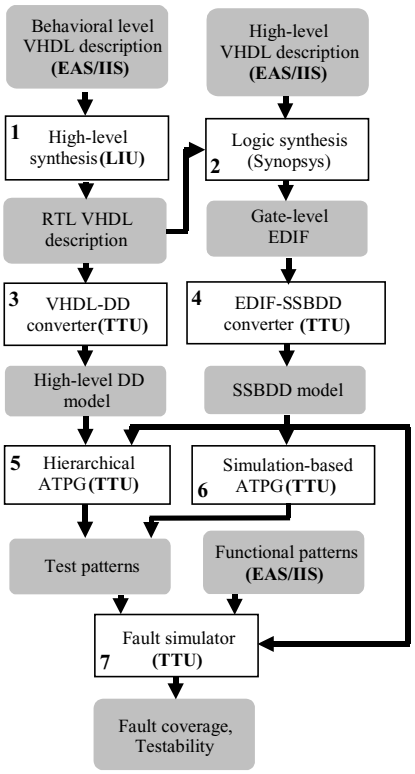
MOSCITO is an Internet based platform for combining design tools via TCP/IP sockets. The MOSCITO system provides:

- an open interface for tool integration
- mechanisms for a network-wide lookup for searching registered tools, start-up, configuration, initialisation and runtime control
- the communication between tools for data transfer
- an open interface for user defined workflows
- a graphical user interface for controlling the workflow and the remote tools, and for viewing status information and results.

Substantially, MOSCITO is relied on a Client-Server concept, i.e. there are one Master Server, several Slave servers and arbitrary number of clients. The service requested is provided by the Slave servers provided with software modules called Agents. These agents encapsulate service provided by working tools (program executables).

[1] This work was partially supported by EC Copernicus project VILAB CP977133, the Swedish National Board for Industrial and Technical Development (NUTEK), Swedish Academy of Engineering Sciences, and the Estonian Science Foundation (Grants No 3658 and 4003).

An Agent can be seen as an intelligent wrapper around another stand-alone program (written earlier by third party). The Agent is capable to communicate with Servers. Any time, it is possible to add and remove Agents. All Slave servers are registered in the Master Server. That way all Agents (i.e. available services) are also registered in the Master server. The USER accesses first the Master server and will get a list of services provided. After selecting the service (Agent) wanted, the USER is automatically re-directed to the Slave



Server, and work with the tool can start.

Fig.1. Internet-based design and test flows

The detailed description of the MOSCITO environment developed in VILAB is given in [3,4].

2.2. Tool integration

The main features of the tool integration conception are:

- § Each partner's application is embedded into a MOSCITO agent
- § The communication between of the application and the agent is file based.
- § The access to the communication files is handled by adaptation of the agent to absolute file paths.
- § The application invocation is based on command lines. Additional outputs (e.g. error messages) are redirected to log files.
- § The agents are implemented in JAVA and the communication is realized via network sockets.
- § MOSCITO provides additional tools for workflow control and monitoring as well as result visualization.

To validate the MOSCITO system and to collect experiences while using it for real-life applications an experimental tool environment for high-level synthesis and test pattern generation (Fig.1) was developed and mapped to a MOSCITO workflow. In the following the functionality of the tools in this workflow are explained in detail.

3. Tool descriptions

3.1. Interface to system-level HW/SW co-design

The initial entry to high-level design flow is represented by a system-level HW/SW co-design back-end tool developed at TU Darmstadt [5].

Fig.2 shows the overall co-design flow: within the DICE design environment [5] an initial mixed HW/SW specification (mixed VHDL/C description) will be simulated and repartitioned according to predefined design constraints. Before starting hardware synthesis, such parts of the specification, which have been initially assigned to become software (C specification) and were moved into hardware modules during partitioning, have to be migrated to VHDL.

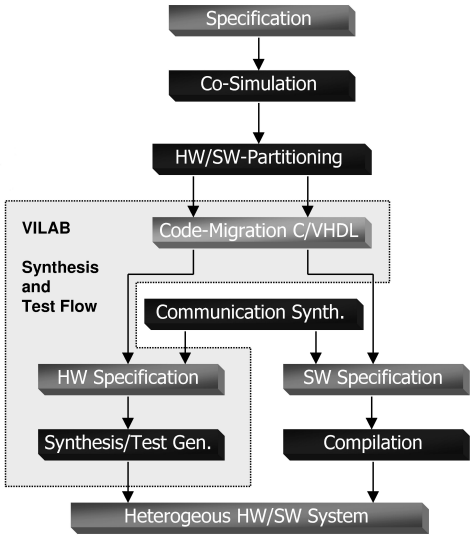


Fig.2. Embedding Design & Test into codesign flow

This migration is performed by a C to VHDL converter, which is connected as a front-end to the integrated design flow. Therefore this converter enables the targeted design flow to process C specifications for hardware synthesis. C processes are parsed and stored in an C Structure Tree (CST) object library. Several code transformations can be applied based on the object structure. A VHDL code generator back-end is generating an VHDL entity/architecture pair, which

contains one or several VHDL processes which serve as the input for high-level synthesis system CAMAD developed at LIU [6].

3.2. High-level synthesis system CAMAD

The CAMAD high-level synthesis system (Block 1 in Fig.1) is built around an internal design representation, called ETPN (Extended Timed Petri Net), which has been developed to capture the intermediate results during the high-level synthesis process. The use of Petri nets provides a natural description of concurrency and synchronization of the operations and processes of a hardware system. It gives thus a natural platform to represent and manipulate concurrent processes of VHDL specifications.

ETPN is used as a *unified* design representation which allows the synthesis algorithm to employ an iterative improvement approach to carry out the synthesis task. The basic idea is that once the VHDL specification is translated into the initial design representation, it can be viewed as a primitive implementation. Correctness-preserving *transformations* can then be used to successively transform the initial design into an efficient implementation. CAMAD *integrates* the operation scheduling, data path allocation, control allocation and, to some degree, module binding sub-tasks of high-level synthesis. This is achieved by developing a set of basic transformations of the design representation which deals *simultaneously* with partial scheduling and local data path/control allocation. An optimization algorithm is then used to analyze the (global) design and select transformations during each step of the iterative improvement process.

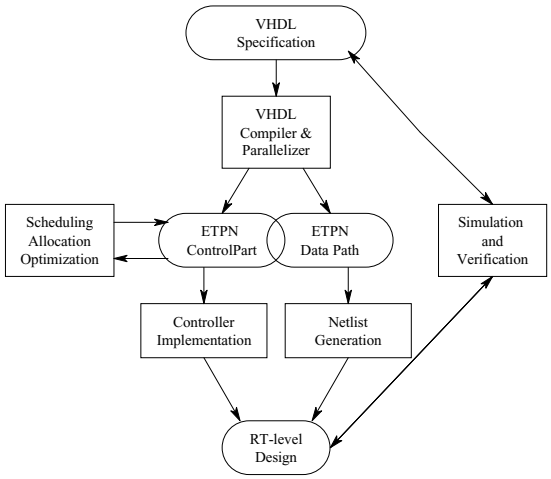


Fig.4. Overview of CAMAD

Fig. 4 illustrates the basic structure of the CAMAD. As the output of CAMAD, a RTL implementation in structural VHDL is generated which consists of a data path netlist and a controller. An interface (Block 3 in Fig.1) has been built between CAMAD and hierarchical ATPG (Block 5) developed at TTU [7].

3.3. Hierarchical ATPG DECIDER

A hierarchical test generation system DECIDER (Block 5 in Fig.1) has been developed which includes a Register-Transfer Level (RTL) VHDL interface for importing high-level design information, and also an EDIF interface (Block 4) for importing gate-level descriptions of logic. The structure of DECIDER is presented in Fig.5.

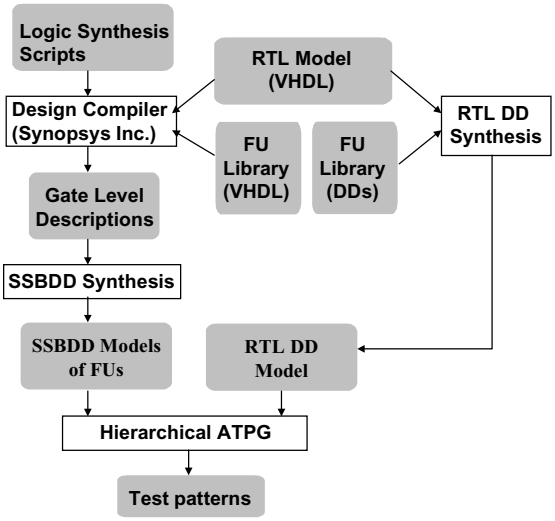


Fig.5. Overview of DECIDER

The ATPG uses a top-down approach, with a novel method of combining random and deterministic techniques. Tests are generated for each functional unit (FU) of the system separately. First, a high-level symbolic test frame (test plan) is created for testing the given FU by deterministic search. As the result, a symbolic path for propagating faults through the network of components is activated and corresponding constraints are extracted. The frame will adopt the role of a filter between the random TPG and the FU under test. If the filter does not allow to find a random test with 100% fault coverage for the component under test, another test frame will be chosen or generated in addition to the previously created ones. In such a way, the following main parts in the ATPG are used alternatively: deterministic high-level test frame generator, random low-level test generator, high-level simulator for transporting random patterns to the component under test and low-level fault simulator for estimating the quality of random patterns.

4. Experimental research

The described environment has been tested in the frame of European project VILAB by the partners for

several designs The following three subflows may be carried out (the reference to the exploited tools in Fig. 1 is given in parentheses):

- § design with hierarchical and deterministic test generation (using the blocks 1,2,3,4,7);
- § design with low-level simulation-based test generator (2,4,6,7);
- § design with user defined functional tests (2,4,7).

The first flow is used in connection with the high-level synthesis tool CAMAD (1) developed by LIU. The output of the synthesis tool will be a RTL description presented in a subset of VHDL which is converted (3) into the input of the hierarchical test generator (5). If the design is presented in a VHDL description not corresponding to the defined VHDL subset, this flow with hierarchical test generation cannot be used. Instead of that, the second flow based on simulation-based test generator (6) is suggested. If the fault coverage of user functional tests is sufficient (determined by fault simulation using blocks 2,4,7), no additional test generation is needed (third flow).

	DECIDER		GATEST		HITEC	
	Fault cover %	Time s	Fault cover %	Time s	Fault cover %	Time s
GCD	91.0	3.4	92.2	89.8	89.3	195.6
Mult 8x8	79.4	13.6	77.3	1585	63.5	1793
Diffeg	95.8	15.8	96.0	9720	95.1	N.A.

Table 1. Experimental results for hierarchical ATPG

This environment has been utilized for research purposes. For example, the performance of the hierarchical test generator DECIDER (7) was compared against the known tools GATEST [8] and HITEC [9]. The results of comparison of ATPGs are given in Table 1 which show the excellent performance of DECIDER.

5. Summary

In the paper an Internet-based environment for high-level design and test generation for digital systems is presented. The environment is focused on supporting SW/HW codesign, high-level and logic design flows with test pattern generation and fault simulation at register-transfer and gate level operational activities. The system provides interfaces and links to commercial design environments. The functionality of the integrated system was verified by several benchmark circuits and by different design and test flows.

The scientific mission of joining partners' competence by a close cooperation in a virtual environment to create synergy has been achieved. Four partners formed a virtual laboratory and joined their competences and know-how in design methodology (IIS/EAS, THD), high-level synthesis and testability (LIU) and hierarchical test generation (TTU) to achieve a new quality in research by remotely carrying out joint design and test flows. Including new original test oriented operations into a commercial design environment and design flows helps to increase the testability of designs and in this way to reach higher quality of products. The convenience of intervening test oriented tools into the design process contributes to speeding up the whole design process and to reducing time-to-market.

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

References

[1] P.Schneider, S.Parodat, A.Schneider, P.Schwarz: A modular approach for simulation-based optimization of MEMS. *Design, Modeling, and Simulation in Microelectronics*, 28-30 November 2000, Singapore, pp 71-82, SPIE Proceedings Series Volume 4228.

[2] MOSCITO: http://www.eas.iis.fhg.de/solutions/moscito.

[3] A.Schneider, E.Ivask, P.Miklos, J.Raik, K.-H.Diener, R.Ubar, T.Cibakova, E.Gramatova. Internet-based Collaborative Test Generation with MOSCITO. *Proc. of DATE'02*, Paris, France, March 4-8, 2002, pp.221-226.

[4] A. Schneider, K.-H. Diener, J.Raik, R.Ubar, E. Gramatova, M.Fisherova, W.Pleskacz, W.Kuzmicz. Defect-Oriented Test Generation and Fault Simulation in the Environment of MOSCITO. In this proceedings.

[5] T.Hollstein, J.Becker, A.Kirschbaum, M.Glesner. HiPART: A New Hierarchical Semi-Interactive HW/SW Partitioning Approach with fast Debugging for Real-Time Embedded Systems. CODES/CASHE'98.

[6] G.Jervan, P.Eles, Z.Peng, J.Raik, R.Ubar. High-Level Test Synthesis with Hierarchical Test Generation. *17th NORCHIP Conf.*, Oslo, Nov. 8-9, 1999, pp.291-296.

[7] J.Raik, R.Ubar: Fast Test Pattern Generation for Sequential Circuits Using DD Representations. *J. of Electronic Testing: Theory and Applications. Kluwer Acad. Publishers*. Vol. 16, No. 3, pp. 213-226, 2000.

[8] E.M.Rudnick, J.Patel, G.S.Greenstein, T.M.Niermann: Sequential Circuit Test Generation in a Genetic Algorithm framework. *DAC.,* pp. 698-704, 1994.

[9] M.Niermann, J.H.Patel: HITEC: A Test Generation Package for Sequential Circuits. *European Conf. Design Automation*, pp. 214-218, 1991.