Internet–based Testability Driven Test Generation in the Virtual Environment MOSCITO

A. Schneider, K.-H. Diener, G. Elst
Fraunhofer Institute for Integrated Circuits
Branch Lab Design Automation (Germany)

E. Ivask, J. Raik, R. Ubar
Tallinn Technical University (Estonia)
Outline

Introduction and motivation:
Test generation and fault simulation
variety of tools, complex workflow

What are the problems?
platforms, performance, installation and administration

Our solution: MOSCITO
basic idea, fundamental concept, implementation, firewall traversal

Current Results: Internet-based work with MOSCITO
Test generation workflow works between different European cities

Summary
Introduction and motivation:
Test generation and fault simulation
variety of tools, complex workflow

What are the problems?
platforms, performance, installation and administration

Our solution: MOSCITO
basic idea, fundamental concept, implementation, firewall traversal

Current Results: Internet-based work with MOSCITO
Test generation workflow works between different European cities

Summary
Starting Point: SoC Design Flow

Synthesis of digital systems (SoC, ...) with CAD tools (e.g. Synopsys)
Starting Point: SoC Design Flow

Synthesis of digital systems (SoC, ...) with CAD tools (e.g. Synopsys)

System on chip FPGA on board
Test pattern generation and fault simulation play an important role in SoC design.

FPGA (System) on chip or on board

Synthesis of digital systems (SoC, ...) with CAD tools (e.g. Synopsys)
VHDL (behavioral) → C-VHDL translator (TUD) → VHDL (behavioral) → High-level synthesis

High-level synthesis → Logic synthesis → Schematic entry

Logic synthesis → RTL VHDL → VHDL-DD translator (LIU, TTU) → High-level DD model

RTL VHDL → Gate-level EDIF → EDIF-ISCAS translator (TTU) → ISCAS netlist

Gate-level EDIF → EDIF-SSBDD translator (TTU) → ISCAS-SSBDD translator (TTU) → ISCAS Benchmarks

EDIF-ISCAS translator (TTU) → Commercial CAD software

VHDL (high level) → Logic synthesis → Schematic entry

VHDL (behavrial) → Commercial CAD software Benchmarks

VHDL (high level) → Schematic entry

VHDL (behavioral) → Commercial CAD software

VHDL (behavioral) → Commercial CAD software

VHDL (behavioral) → Commercial CAD software

VHDL (behavioral) → Commercial CAD software
Turbo Tester - A Set of Low-level ATPG Tools

- Turbo Tester tools operate on the SSBDD model (Structurally Synthesized Binary Decision Diagrams)

- set of tools for Automatic Test Pattern Generation (ATPG), Built-In Self-Test (BIST) emulation, fault simulation, test set optimization and multi-valued simulation
Outline

Introduction and motivation:
Test generation and fault simulation
variety of tools, complex workflow

What are the problems?
platforms, performance, installation and administration

Our solution: MOSCITO
basic idea, fundamental concept, implementation, firewall traversal

Current Results: Internet-based work with MOSCITO
Test generation workflow works between different European cities

Summary
What are the problems?

Many tools from industry and academia are in use.

- Synopsys
  - Synthesis
  - Test Pattern Generation
  - TurboTester (Tallinn)

- Test Pattern Generation
  - DefGen (Bratislava)

- Translators
  - EDIF-SSBDD (Tallinn)

- Other commercial design tools ...
What are the problems?

Many tools from industry and academia are in use.

Installation and administration cost a lot of man power.
What are the problems?

Many tools from industry and academia are in use.

Installation and administration cost a lot of man power.

Time and money are wasted for studying documentations and trainings.

- Synopsys
  - Synthesis
  - Test Pattern Generation
  - TurboTester (Tallinn)
- Test Pattern Generation
  - DefGen (Bratislava)
- Translators
  - EDIF-SSBDD (Tallinn)
- other commercial design tools ...
What are the problems?

Several hardware architectures are utilized.
What are the problems?

Several hardware architectures are utilized.

Appropriate resources are required for different design tasks.
What are the problems?

Several hardware architectures are utilized.

Appropriate resources are required for different design tasks.

Tools are running on different operating systems.
Introduction and motivation:
Test generation and fault simulation
variety of tools, complex workflow

What are the problems?
platforms, performance, installation and administration

Our solution: MOSCITO
basic idea, fundamental concept, implementation, firewall traversal

Current Results: Internet-based work with MOSCITO
Test generation workflow works between different European cities

Summary
Basic Idea of MOSCITO

Available tools are applied.

- EDIF2SSBDD
- TurboTester
- Tst2Alb
- ALB
- DefGen
- EDIF2ISCAS
Basic Idea of MOSCITO

Available tools are applied.

Tools can be assembled into workflows as requested.
Basic Idea of MOSCITO

Available tools are applied.

Tools can be assembled into workflows as requested.

Coupling of different tools via Internet.
Basic Idea of MOSCITO

Available tools are applied.

Tools can be assembled into workflows as requested.

Coupling of different tools via Internet.

MOSCITO front end allows controlling the workflow and visualization of results.
Fundamental Concept
Fundamental Concept

Each tool has to be encapsulated.
Fundamental Concept

Each tool has to be encapsulated.

MOSCITO Agent provides a uniform interface for communication.
Each tool has to be encapsulated.

MOSCITO Agent provides a uniform interface for communication.

MOSCITO infrastructure implements data transfer via TCP/IP sockets.
Fundamental Concept

Each tool has to be encapsulated.

MOSCITO Agent provides a uniform interface for communication.

MOSCITO infrastructure implements data transfer via TCP/IP sockets.

Data exchange is based on XML and Java serialization.
Each tool has to be encapsulated.

MOSCITO Agent provides a uniform interface for communication.

MOSCITO infrastructure implements data transfer via TCP/IP sockets.

Data exchange is based on XML and Java serialization.
Tool Encapsulation

The agent receives the input data and passes them to the embedded tool.
Tool Encapsulation

The agent receives the input data and passes them to the embedded tool.

Agent starts the tool and waits for result data.

MOSCITO Agent

control information from user’s front end

input data

result data

file

file
Tool Encapsulation

The agent receives the input data and passes them to the embedded tool.

Agent starts the tool and waits for result data.

After conversion/serialization the result data will be sent to the following agent.
Tool Encapsulation

The agent receives the input data and passes them to the embedded tool.

Agent starts the tool and waits for result data.

After conversion/serialization the result data will be sent to the following agent.

```
MOSCITO Agent

pipe: stdin
input data

pipe: stdout
result data

conversion

serialization

control information from user’s front end
```
Tool Encapsulation

The agent receives the input data and passes them to the embedded tool.

Agent starts the tool and waits for result data.

After conversion/serialization the result data will be sent to the following agent.
Tool Encapsulation

The agent receives the input data and passes them to the embedded tool.

Agent starts the tool and waits for result data.

After conversion/serialization the result data will be sent to the following agent.
Communication between Firewall-protected sites

All MOSCITO agents and the graphical front-end program communicate via TCP/IP.
Communication between Firewall-protected sites

All MOSCITO agents and the graphical front-end program communicate via TCP/IP.

Problem is that firewalls do not allow direct communication connections.
Communication between Firewall-protected sites

All MOSCITO agents and the graphical front-end program communicate via TCP/IP.

Problem is that firewalls do not allow direct communication connections.

It makes no sense to open a few ports and to make a firewall more transparent.
All MOSCITO agents and the graphical front-end program communicate via TCP/IP.

Problem is that firewalls do not allow direct communication connections.

It makes no sense to open a few ports and to make a firewall more transparent.

There are two solutions for the problem:
- WWW server (HTTP)
- Proxy (SOCKS)
All MOSCITO agents and the graphical front-end program communicate via TCP/IP.

Problem is that firewalls do not allow direct communication connections.

It makes no sense to open a few ports and to make a firewall more transparent.

There are two solutions for the problem:
- WWW server (HTTP)
- Proxy (SOCKS)

Currently we are implementing and testing the proxy-based solution.
Software Architecture

MOSCITO is implemented in Java. (Sun J2SE 1.4.x)
Software Architecture

MOSCITO is implemented in Java. (Sun J2SE 1.4.x)

Kernel provides basic functionality.
Software Architecture

MOSCITO is implemented in Java. (Sun J2SE 1.4.x)

Kernel provides basic functionality.

Using the open interfaces (API) users can add new components (agents, workflows, viewers).

MOSCITO Kernel

Operating system with Java Virtual Machine (Windows, Solaris, Linux, ...)

Workflow
Agent
Viewer
Scope
Desktop
MOSCITO is implemented in Java. (Sun J2SE 1.4.x)

Kernel provides basic functionality.

Using the open interfaces (API) users can add new components (agents, workflows, viewers).
MOSCITO is implemented in Java. (Sun J2SE 1.4.x)

Kernel provides basic functionality.

Using the open interfaces (API) users can add new components (agents, workflows, viewers).
MOSCITO is implemented in Java. (Sun J2SE 1.4.x)

Kernel provides basic functionality.

Using the open interfaces (API) users can add new components (agents, workflows, viewers).
Outline

Introduction and motivation:
Test generation and fault simulation
variety of tools, complex workflow

What are the problems?
platforms, performance, installation and administration

Our solution: MOSCITO
basic idea, fundamental concept, implementation, firewall traversal

Current Results: Internet-based work with MOSCITO
Test generation workflow works between different European cities

Summary
Internet-based Workflow between Tallinn, Dresden and other sites

Internet-based collaborative test generation between partners from different European cities.

All partners have to maintain their own tools.

Each partner can profit from all tools within the needed workflow.
MOSCITO
Front End
Front End

- Browser
- Workflows
- Agents
Front End

Browser
- Workflows
- Agents

Scope
- Chart
- Text
- Image
Front End

Browser
- Workflows
- Agents

Scope
- Chart
- Text
- Image

Console
- Infos
- Errors
All available agents are listed in Browser window.
All available agents are listed in Browser window.

User can select the needed agents and can configure them via dialog windows.
Infos, warnings and error messages will be sent directly to Console window.

User can observe the tool status.
Each tool (agent) sent results to the database of front-end.

User can explore result data with Scope window.

Results can be stored to local file system.
Each tool (agent) sent results to the database of front-end.

User can explore result data with Scope window.

Results can be stored to local file system.
Summary

Test generation is based on various tools (pattern generators, fault simulators, translators).

Tools run on different platforms. Appropriate resources are needed.

MOSCITO provides an infrastructure for an Internet-based tool integration. Partners are enabled to cooperate remotely.

The maintenance costs for users are minimal. Test generation is available as a set of services.

Users with minimal knowledge in test engineering and tools involved there, can carry out complex test generation tasks.

MOSCITO Front End allows to observe the state of flow execution and results (files, images, numerical data).
Further information

http://www.eas.iis.fhg.de/solutions/moscito

http://www.pld.ttu.ee/tt
Other Approaches and Architectures

CORBA
  • very complex

Java RMI
  • could be used for communication and naming service

COM/DCOM
  • available for Microsoft platform only

.NET
  • under development

HLA
  • targeted to simulation

TENT (DLR)
  • similar to MOSCITO approach

Lavana et al
  • fine-grained workflow concept

Chamois (Won Kim et al)
  • ...

James (Uhrmacher)
  • targeted to simulation of Multi Agent Systems

Astai(R) (C-Lab)
  • CORBA-based