CROSS-TOOL DEVELOPMENT AND VERIFICATION OF

Hardware Designs

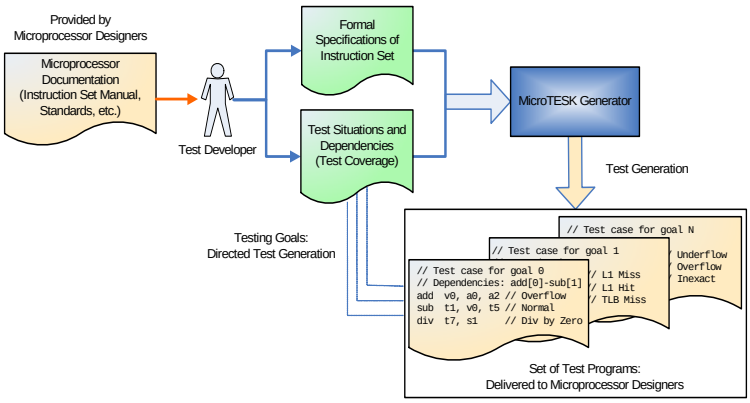
SPECIFICATION-DRIVEN APPROACH

TRUTH IN CONFORMANCE

MicroTESK: Test Program Generation for Microprocessors

MicroTESK is a technology to generate test programs in assembler language for microprocessors and other programmable devices such as arithmetical coprocessors, controllers, etc. MicroTESK significantly improves the quality of verification by systematic construction of various test cases based on light-weight formal specifications of microprocessor instruction set and testing goals in terms of instruction-level test coverage. A set of special test data generation libraries (for integer and floating-point arithmetic, memory management, etc.) simplifies development of test generators and reduces labor cost of microprocessor verification.

Test Development Process



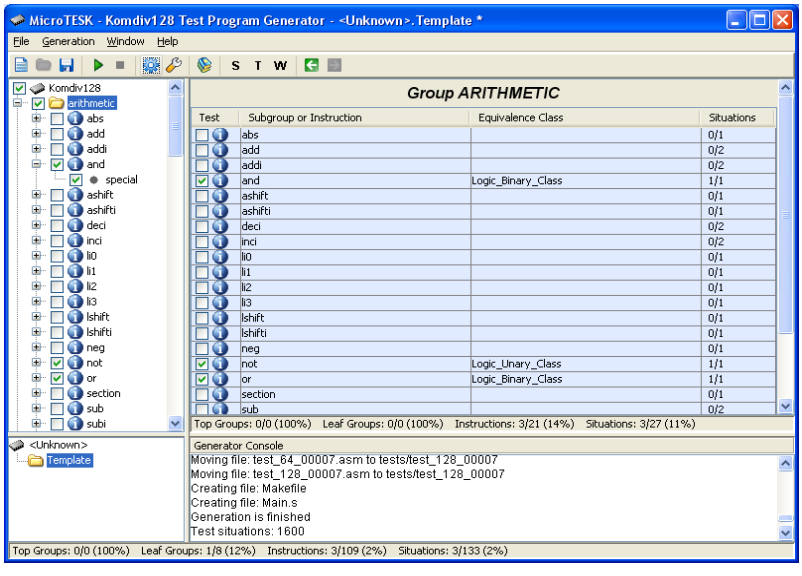
Microprocessor designers provide test developers with documentation on the target microprocessor: instruction set manual, standards, etc. Test developers create formal specifications and define test coverage goals in the form of test situations (arithmetical exceptions, cache hit/misses, and other events) and dependencies between instructions (via registers or memory). Both descriptions are used as inputs for the MicroTESK Generator. The generator automatically builds a set of test programs that cover all specified testing goals.

MicroTESK Features

- *Directed test generation* – test programs are generated according to specified goals
- *High automation* – technology provides high level of automation
- *Self-checking tests* – test programs contain checks of microprocessor behavior
- *Java language* – formal specifications are developed in widely-adopted Java
- *Generation libraries* – there are ready-to-use test generators for typical instructions
- *Graphical interface* – generator has graphical user interface

MicroTESK Generator

The main components of the MicroTESK Generator that are responsible for construction of test cases form a generator core. Apart from the core components, the generator has a number of libraries which simplify development of microprocessor specifications and include many ready-to-use components like iterators, test data generators, etc. To facilitate using the MicroTESK Generator, it has graphical user interface.



Institute for System Programming
Russian Academy of Sciences
25, A. Solzhenitsyn Street,
Moscow, 109004, Russia

Phone: +7 (495) 912 5317
Fax: +7 (495) 912 1524
E-mail: hardware@ispras.ru
http://hardware.ispras.ru

Institute for System Programming of RAS

Presented tools and technologies were developed at the Software Engineering Department of the Institute for System Programming, Russian Academy of Sciences (ISPRAS). The Institute performs both academic research and industrial development projects as well as provides advanced services and consulting in various areas of software engineering, information technologies and computer science.

One of the main directions of the Software Engineering department is advanced verification and testing of software and hardware systems. Since 1994, experts of the department have been participating in projects on automated test development for complex industrial software and hardware under cooperation with such companies as Nortel Networks, Microsoft, Intel, VIA Technologies, Linux Foundation, NIISI RAS, Luxoft, VimpelCom, etc. Based on the experience gained from these and other projects, experts of the department created a number of innovative methods and corresponding technologies and tools for automated development of various types of tests, which are united under UniTESK technology stack. These methods, technologies and tools have proved their efficiency for formal specification and testing of operating systems, compilers, implementations of telecommunication protocols, HDL hardware models, and other types of systems. While we always take opportunities to further improve and enrich them using feedback from new on-going projects.

CROSS-TOOL DEVELOPMENT AND VERIFICATION OF

Hardware Designs

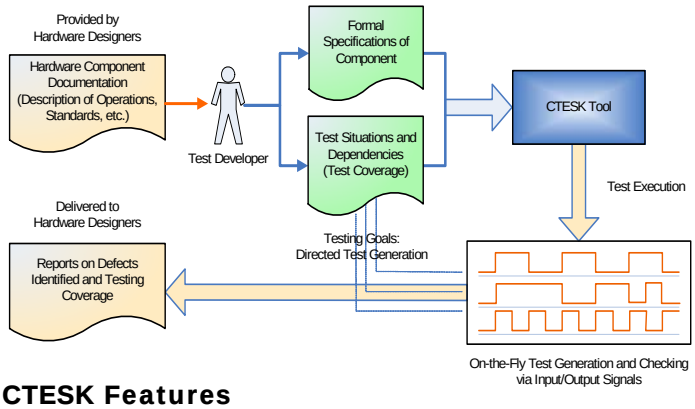
SPECIFICATION-DRIVEN APPROACH

TRUTH IN CONFORMANCE

UniTESK: Simulation-Based Verification of HDL Models

UniTESK approach to hardware component verification is based on formal specifications of component's operations. Formal specification slips each operation into a sequence of microoperations corresponding to separate stages and defines each microoperation in the form of pre- and post-conditions. This approach provides cycle-accurate component specification. On the base of such specifications the CTESK tool can automatically generate test sequences, verify correctness of component behavior in response to test action, and estimate verification completeness. CTESK implements advanced methods of test sequence generation. Formal specifications of a component are interpreted as finite state machine (FSM) of a special type. Test sequence is constructed on-the-fly by traversing FSM state graph. A set of special test data generation libraries (for integer and floating-point arithmetic, etc.) simplifies development of test generators and reduces labor cost of unit-level verification.

Test Development Process



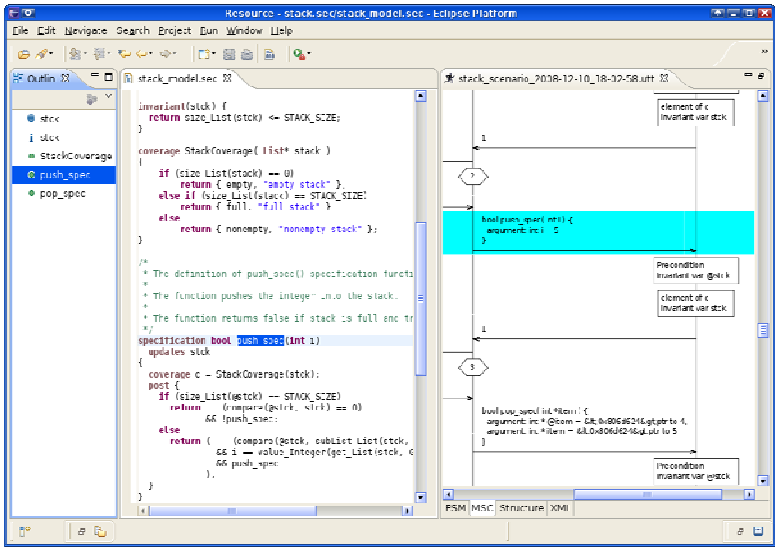
Hardware designers provide test developers with documentation on the target hardware component: description of operations, standards, etc. Test developers create formal specifications and define test coverage in the form of test situations (arithmetical exceptions, cache hit/misses, and other events) and dependencies between operations (via registers or memory). Both descriptions are used as inputs for the CTESK tool. Test sequence generation is performed automatically on-the-fly in an HDL simulator. The result of test execution is a detailed report on defects identified and testing coverage achieved during testing.

CTESK Features

- *Directed test generation* – test sequences are generated according to specified goals
- *High automation* – technology provides high level of automation
- *Self-checking tests* – tests automatically perform checks of design behavior
- *SeC language* – formal specifications are developed in SeC language
- *Generation libraries* – there are ready-to-use test generators for typical operations
- *Trace tools* – CTESK has advanced tools for test result analysis

CTESK Tool

CTESK test development tool is an implementation of the UniTESK technology for the C programming language. It uses special language SeC (Specification extension of C) to develop testbench components. Originally CTESK was developed for testing safety-critical C software like operating systems and implementations of telecommunication protocols. By now, we have adapted CTESK for simulation-based verification of HDL models.



Institute for System Programming
Russian Academy of Sciences
25, A. Solzhenitsyn Street,
Moscow, 109004, Russia

Phone: +7 (495) 912 5317
Fax: +7 (495) 912 1524
E-mail: hardware@ispras.ru
http://hardware.ispras.ru

Institute for System Programming of RAS

Presented tools and technologies were developed at the Software Engineering Department of the Institute for System Programming, Russian Academy of Sciences (ISPRAS). The institute performs both academic research and industrial development projects as well as provides advanced services and consulting in various areas of software engineering, information technologies and computer science.

One of the main directions of the Software Engineering department is advanced verification and testing of software and hardware systems. Since 1994, experts of the department have been participating in projects on automated test development for complex industrial software and hardware under cooperation with such companies as Nortel Networks, Microsoft, Intel, VIA Technologies, Linux Foundation, NIISI RAS, Luxoft, VimpelCom, etc. Based on the experience gained from these and other projects, experts of the department created a number of innovative methods and corresponding technologies and tools for automated development of various types of tests, which are united under UniTESK technology stack. These methods, technologies and tools have proved their efficiency for formal specification and testing of operating systems, compilers, implementations of telecommunication protocols, HDL hardware models, and other types of systems. While we always take opportunities to further improve and enrich them using feedback from new on-going projects.