Some Results in Automatic Functional Test Design for VLSI Circuits

Ján Hudec
Slovak University of Technology
Institute of Computer Systems and Computer Networks, Faculty of Informatics and Information Technologies, Ilkovičova 3, 842 16 Bratislava, Slovak Republic
jhudec@fiit.stuba.sk

Abstract. This paper deals with the design and implementation of a universal functional test generator for VLSI circuits. Our approach to test generation - the functional test generation method - is based on knowledges and functional description of VLSI systems at functional VHDL level and the algorithm for automatic generation of test (normal executable test sequence of instructions) and arrangement, is used in very flexible and effective tool - functional test generation software system. Also a short methodology overview for the test synthesis of VLSI and ASIC circuits using an automated process of the VHDL synthesis simultaneously with Automatic Functional Test Generator (AFTG) is presented. The determination of the test efficiency of instructions mixes is discussed.

Keywords. VHDL synthesis, VLSI functional test generation, test synthesis, and verification.

1. Introduction

As a consequence of exploiting the ever more advanced technologies, the complexities of electronic-based products have increased significantly and there is a great need to incorporate design for testability and test generation into the development of these products. One from many important problems in this area is to create a powerful tool for automatic test design of digital systems, which supports automated test pattern generation. In recent years, functional testing has been presented as a suitable solution to the problem of test generation for complex VLSI systems, such as microprocessors and specially designed ASIC systems. In this area, a number of functional level test approaches for VLSI systems have been proposed.

The AFTG (automatic functional test generator) project provides a consistent set of tools for definition of a test method for complex programmable systems, functional test generation, test verification, determination of test efficiency, fault simulation and testability analysis.

2. Testbench and methodology of test synthesis, simulation and verification

Once the designer has described the design, the design must be verified to check that the specification has been followed. The commonest verification method is to apply input stimuli signals during a simulation and then read the outputs signals from the system design. A major disadvantage of using the simulator’s input stimuli language is that it depends from simulator.

Another method of verification is to write the test model generation and the output signals check in VHDL. This means that you design a testbench which both provides input signals and tests the output signals from the design. The recommendation is to use both the VHDL testbench and to verify the components using system simulation.

The major advantages of the VHDL testbench are its speed and the fact that it is platform-independent, as it is described in VHDL language.

The disadvantage is that the VHDL testbench can contain the same logical errors as the design [4]. For our purpose, it may be beneficial to build a test environment in full or in part in specific VHDL functional diagnostics tool.

Naturally, the basic question is how effective test programs (testing mix – is the meaningful sequence of instructions) can be created automatically and what AFTG design rules must be followed in VHDL code in order to make this possible.

Our methodology - the AFTG approach to functional testing is based upon the following principles:
• Use of an identification test method (no extra fault model of system is needed),
• Use of a system for the automated generation of test programs for actual VLSI systems.

We consider this approach as most effective mainly because it is used the knowledge-based method that is very close to the identification test method and is a combination of heuristic and random pseudotrivial approach [2].

The architecture and behaviour of the system is described in VHDL language divided into some basic parts (such as ALU, memory, control unit, I/O unit, block of registers, etc.).

In addition to generating instruction mixes we developed method, in which we used:
• Solid state choice of instructions,
• Standard computer system random number generator, which is capable to fetch instructions from appropriate groups using choice algorithm and to add suitable operands.

Such a way, we obtained that the ability of system’s testing is the property of generated testing mix. We concentrated our effort on this idea. The methodology (called Grouping) of test synthesis was briefly introduced in [1], [2], [3] and the procedure of obtaining of testing mixes consists from following steps:
1. Analysing and partition of instruction set to groups (G - number of groups).
2. Establishing number of instruction in each group (N).
3. Random generator RNG generates a number ni from 1 to Ni (Ni is maximal number of instruction in each group i).

This is a pointer to instruction in actual group of instruction. Number of register combinations for each instruction is Q.

OPER is field of pointers for estimating of access index to field with appropriate operands.
4. For each fetch instruction in actual group, random generator RNG generates number from interval 1 to Q for establishing pointer OPER for access to field with operand for this instruction.
5. Repeat this procedure for desired number of instruction in testing mix.

System AFTG (Fig. 1) generates a test programs (program composed from various parts of testing mixes) in assembly language according to the user’s choice of:
• Test generation option (used method, number and type of mixes),
• VLSI system test (number of blocks, instructions), using the reported methodology.

AFTG provides test programs in assembly language, which are easy to use during diagnosis experiments (leads to readable programs), but AFTG has also a possibility for:
• Generating the executable code,
• Watching the internal states of system and simulating the function.

AFTG fully makes extended arrangement of instructions and automatic test generation of test mixes for whole system.

3. Some implementation results and test verification

The whole AFTG system is implemented in C# language in environment of Visual Studio.NET. The ModelSim simulator was used due to very good property for incorporate it to the whole AFTG design. The verification of the efficiency of the generated tests of systems (VLSI microprocessor type circuits) is obtained from the diagnostics efficiency tool, which is integrated into the product. The set of tools in AFTG test program generator is open and can be easily extended because of the description parameter’s versality of representing system model.
• Reference system description in VHDL (i.e. processor),
• Various descriptions of systems in VHDL with fault function assumption,
• Method of composition of testing mixes (i.e. the number of instructions in each testing mix),
• Validation by simulation process (what kind of comparison),
• Test efficiency result.

Some previews from AFTG system menu graphic interface are in Fig. 2, 3, 4.
We concentrated on the comparative method of the test evaluation (comparison with a reference ("gold") system) in two ways:
• Comparison of the memory contents after simulation of the whole testing mix,
• Comparison of simulation runs that means comparison of registers of processors after each instruction of testing mix.

The part of the whole VHDL-based design was the automatic functional test generation to several systems, such as family iAPX 86 microprocessor (from 8086 to 80486 - Intel, AMD), Pentium, Motorola 68000 family (68020, 68030, 68LC040), neural TNP processor and hypothetical 32-bit DP32 processor.

Some results of test efficiency verification of iAPX86 system - Ten functional errors are modelled in processor: Stuck-at- zero/one on zero bit of address bus or data bus, zero bit of address bus is equal to first bit of data bus, register AX have zero value and some errors in execution of instructions MOV, STC, CWD, NOT.

The number of detected faulty processors with 15 testing mixes composed from 6 instructions is shown from Fig. 5. It was 9 processors from 10. We have used the comparison method of registers contents of reference and faulty processor after each instruction of testing mix.

The validation result from 30 testing mixes composed from 72 instructions is on the Fig. 6. The number of detected faulty processors in this particular arrangement was nearly 10, i.e. 100 %.

Based on the simulation process validation we affirm that the longer testing mix gives high efficiency in detection of faulty systems. The
major effort in preparing each system test comprised its precise system description, while the realization was automated by AFTG components of whole VHDL synthesis system. After these practical experiences we implemented the sophisticated module for extended test efficiency analysis, which we first used for the DP32 processor (Fig. 7, 8).

The extended analysis compares the difference in longitude of compared files, the difference in number of characters and the average of these values. Additional information about the simulation process is also displayed.

4. Conclusion

On the basis the above specifications and methodologies we developed and implemented a system with computer-aided tools for system synthesis, simulation, fault simulation, verification of efficiency of testing mixes together with solving different test design tasks. The AFTG test design software for systems described at the VHDL level consists of tools for circuit description, multi-valued simulation and testability analysis, test mix generation and test efficiency verification. On higher level the AFTG system supports design and synthesis from various CAD environments, including those of OrCAD, Synopsys, Viewlogic, Mentor Graphics FPGA Advantage and Xilinx Design Suite Technology. Such a way, the CAD synthesis system with AFTG forms the basis of a VHDL functional diagnostics tool (VFDT). AFTG with supporting synthesis CAD tools is a flexible and versatile tool for automated design of comprehensive test programs for complex VLSI circuits, such as intelligent chips, embedded cores, dedicated and specific microprocessors, and ASICs.

5. Acknowledgements

The support by Slovak Science Grant Agency (VEGA 1/0649/09 “Security and reliability in distributed computer systems and mobile computer networks”) is gratefully acknowledged.

6. References