Software-Based Self-Testing of Microprocessors by Exploiting a Virtual Scan Path

Giorgos Dimitrakopoulos, Xrisovalantis Kavousianos, and Dimitris Nikolos

Computer Engineering and Informatics Dept., University of Patras, Greece
{dimitrak, kavousia}@ceid.upatras.gr nikolosd@cti.gr

Abstract. A systematic methodology for generating software-based self-tests for microprocessor cores that can be applied at-speed is introduced in this paper. The test programs emulate the functionality of a scan-path design and their generation is based only on a RT-level VHDL description of the microprocessor.

1 Introduction

Modern GHz microprocessors impose significant challenges to the test community, due to their high complexity and heterogeneity. An alternative to hardware-based self-testing is software-based self-testing, which involves the testing of a microprocessor using its instruction set [1]. The main benefit of software-based self test is that it can be applied in the normal operation mode of the microprocessor, thus applying the required tests at-speed. Additionally, software-based self-testing does not require any design changes neither the insertion of any additional test hardware, and its efficiency has been proven both for stuck-at [1] and delay faults [2]. In this paper a new methodology for software test program generation is introduced. The approach followed for the generation of software-based structural tests does not require any low-level implementation information of the microprocessor, as needed in [1], and it is only based on its RT-level description and its instruction set architecture.

2 The Proposed Methodology

In a non-pipelined microprocessor or microcontroller the existent registers, such as the Instruction Register (IR), the Program Counter (PC), and the Accumulator (ACC), can be set to specific values via an instruction or a certain sequence of instructions assuming certain values for their operands. For example the value of the ACC can be directly set via a load-accumulator instruction and its contents can be observed by applying a store-accumulator instruction. In a similar manner the contents of the PC can be controlled by a jump instruction to a certain memory address.

The scan chain design-for-testability technique allows the values of the registers of the circuit to be directly controlled and observed through the scan input and output pins respectively, in a bit-serial manner.

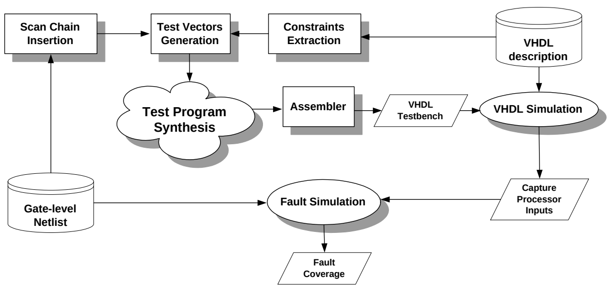

Fig. 1. The proposed flow for the generation of software-based test programs.

Therefore the aim of the proposed methodology is at first to generate test vectors as if the microprocessor employed a scan chain and second to generate the proper sequence of instructions that would emulate the functionality of the scan chain to the *non-scan* microprocessor. Note that the microprocessor does not contain any scan chain. The scan chain insertion takes place only for test pattern derivation purposes by the ATPG. In order the generated ATPG vectors to be *realizable* by processor's instructions, certain constraints are applied to the ATPG tool. For example the nominal values assumed by the IR are limited to the opcodes of the processor's instructions.

The proposed software-based test generation flow is shown in Fig. 1. The main step of the proposed methodology is the test program synthesis procedure, which generates an assembly program that contains the instructions sequences needed to emulate each test vector. In general the sequence of instructions used to emulate the application of one test vector to the microprocessor, assuming a virtual scan path, consists of 6–8 instructions, which are divided into three groups. The first group sets the proper values to the registers, the second applies the virtual scan test vector, and the third group of instructions writes back to memory the new state of the microprocessor. Proper care is taken in order the values of certain registers not to change until their values are observed, that is their contents are written back to memory in order to be compared with the pre-computed values of the non-fault behaviour. During the test application phase the instruction groups lie in specific memory locations. In case that more than one test vectors impose their corresponding instruction sequences to lie in the same memory area, then a conflict is caused. Hence, the conflicting instruction groups are scheduled to different test sessions, i.e. different assembly programs.

The proposed methodology was applied to the PARWAN microprocessor [1] achieving 82% fault coverage after applying an automatically generated test program that consisted of 262 instructions, which were executed in 984 cycles.

References

1. L. Chen, S. Dey: Software-Based Self-Testing Methodology for Processor Cores. IEEE Trans. Computer-Aided Design, Vol. 20, No.3, pp. 369–380, March 2001.
2. W. C. Lai, *et al.*: On testing the path delay faults of a microprocessor using its instruction set. Proc. 18th VLSI Test Symp., pp. 15-20, May 2000.