# New Evolutionary Techniques for Test-Program Generation for Complex Microprocessor Cores

E. Sanchez, M. Schillaci, M. Sonza Reorda, G. Squillero, L. Sterpone, M. Violante
Politecnico di Torino - Dip. Automatica e Informatica
Cso Duca degli Abruzzi 24
10129 Torino – Italy
+39 011 5647092
{ sanchez, schillaci, sonza, squillero, sterpone, violante } @cad.polito.it

## ABSTRACT
Checking if microprocessor cores are fully functional at the end of the productive process has become a major issue. Traditional functional approaches are not sufficient when considering modern designs. This paper describes new improvements for an existing evolutionary algorithm, called µGP, able to generate Turing-complete programs; these are exploited, along with hardware acceleration techniques, to add content to a qualifying test campaign by automatically generating assembly programs. The approach is suitable for medium-sized processor cores. The experimental evaluation performed on a SPARCv8 clearly shows the potentiality of the approach, and the effectiveness of the enhancements to the evolutionary core.

## Categories and Subject Descriptors
D.1.m [Programming Techniques]: Miscellaneous

## General Terms
Algorithms

## Keywords
Evolutionary algorithms, Automatic test program generation.

## 1.INTRODUCTION
After production, any new microprocessor must be checked by means of a *post-production test***,** to ensure that it is fully functional. Production yield is relatively low, so test is one of the biggest issues in delivering a high performance device today [1].

The new evolutionary improvements are verified against a real problem such as pipeline testability for a medium-sized microprocessor core. The task is to automatically add content to a test suite for a qualifying test campaign. The proposed approach focuses on a hard-to-test structure, not easily addressed by the traditional functional techniques.

Methods for automatically generating test programs may be classified either as "with feedback" or "without feedback".


*GECCO'05,* June 25-29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

Techniques without feedback are essentially pseudo-random approaches able to generate random sequences of instructions fitting some specific constraints. This kind of methodologies has been broadly investigated, for instance in [2, 3, 4]. Feedback-based generators, on the other hand, are usually far more effective, but their computational complexity usually prevents the exploitation of such methodologies even against medium-sized microprocessors.

Stemming from [5], [6] describes a feedback-based approach called µGP. The µGP is an evolutionary approach for generating assembly programs tuned for a specific microprocessor. Even though the evolutionary core has been improved to increase its performance, until now the number of evaluations required in the evolution has prevented the µGP from tackling testability on medium-sized microprocessors. For example, assessing the effectiveness of one test program would require several days on a workstation.

## 2.PROPOSED APPROACH
Being based on the µGP, the proposed methodology can be classified as feedback based; it additionally exploits hardware acceleration techniques.

The basic idea behind our hardware acceleration consists in instrumenting the target microprocessor core to measure its internal activity, and to perform fault simulation. Finally, the design is mapped on a FPGA-based device that emulates the core during program execution [7].

The original µGP is described in [6], so only the latest improvements to the method are described here and will be detailed below.

For this work, self-adaptation of the tournament size for tournament selection has been added to the core, making it an endogenous parameter. The tournament size is increased when the tool is able to obtain significant fitness improvements and it is decreased when gains are small or nonexistent.

The purpose of the second upgrade to the tool, a local mutation operator, is to perform an efficient search of the nearby solution space around a high-fitness individual, thereby enabling faster exploitation of the local maximums.

The latest additions to the tool's capabilities are the aging of the individuals and the user option of choosing the elite population size, defined here as the number of individuals that are not affected by aging. The original core implements a distinctly elitist scheme. This may perform poorly on some class of problems. Since the µGP tool is meant to be useable for generic problem

solving, it makes sense to allow it to use either an elitist scheme or a non-elitist one.

## 3.CASE STUDY

The proposed approach was tested on a SPARCv8: a synthesizable VHDL model of a 32-bit microprocessor conforming to the IEEE-1754 architecture [8]. The microprocessor pipeline is organized in 5 stages: *fetch*, *decode*, *execute*, *memory* and *write back*. The netlist resulting from synthesis numbers about 65,000 gates (without the cache).

An initial test set was carefully written according to the method proposed in [9]. Although it is effective, on some parts of the microprocessor the approach is not sufficient. More specifically, 40% of the stuck-at faults in the 742 flip-flops composing the pipeline registers were not detected by the initial test set. A prototype of the proposed approach was built tackling such faults. The hardware accelerator exploits an FPGA board equipped with a Virtex 2000E device. The whole process takes about 26 hours to complete. For comparison, a random test set has been evaluated.

Table 1 summarizes the results. Fault coverage refers to the faults in the pipeline registers.

**Table 1. Fault Coverage Results**

| Test Set | Fault Coverage [%] | Clock Cycles [#] | Instructions [#] |
|---|---|---|---|
| Traditional | 58.89 | 11,263 | 286 |
| Completed | 100.00 | 15,843 | 402 |
| Rnd_Comp | 78.23 | 12,589 | 341 |

Remarkably, the proposed method is able to improve a typical set of test programs composed of random programs or deterministic ones by better tackling internal complex structures such as the processor pipeline.

Five experiments were performed to evaluate the effect of the proposed improvements to the evolutionary core. The first one has been performed using the previous version of the evolutionary core, and is called the Reference. Three experiments were made to isolate the effects of every single improvement to the tool, and are indicated by Age, Loc and Tau. The final one has been performed implementing all of the improvements in the µGP core, and is shown under the name Complete. All of these experiments use the same evolutionary parameters. The features employed in the five experiments are shown in Table 2.

**Table 2. Experimental Setup**

| Experiment | Elitism | Tournament size self-adaptation | Local search |
|---|---|---|---|
| Reference | strong | no | no |
| Age | relaxed | no | no |
| Loc | strong | no | yes |
| Tau | strong | yes | no |
| Complete | relaxed | yes | yes |

Figure 1 reports the best fitness at every generation in all the experiments.

The Loc experiment, exploiting local mutation, even performs worse than the original evolutionary process. This is a sure sign that the fitness function is a deceptive one. The Age experiment, which employs individual aging, greatly relaxes the elitist scheme, allowing greater freedom in the solutions space search. But, contrary to expectations, the evolutionary process exactly follows the original one. Tournament size self-adaptation allows the tool to avoid getting stuck in a local optimum, so the Tau experiment gives better results than the simple elitism relaxation.

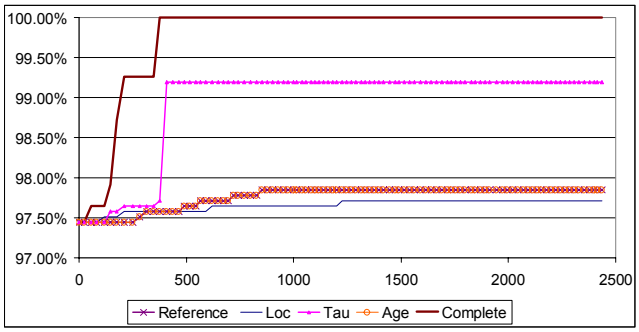


**Figure 1 - Result summary**

## 4.CONCLUSIONS

The paper presents a novel approach for the automatic completion of a test suite for a medium-sized microprocessor core. The technique has been tested on the SPARCv8 as a case study, and experimental results clearly show its effectiveness improving the traditional approach.

The proposed approach introduces different enhancements in µGP, an existing evolutionary algorithm for generating assembly-language programs, and exploits a hardware accelerator to efficiently evaluate individuals. The experimental evaluation shows their usefulness in the specific task.

## 5.REFERENCES


[1] *International Technology Roadmap for Semiconductors*, 2003 edition

[2] K. Batcher, C. Papachristou, "Instruction Randomization Self Test For Processor Cores", *IEEE VLSI Test Symposium*, 1999, pp. 34-40

[3] L. Chen, S. Dey, "DEFUSE: A Deterministic Functional Self-Test Methodology for Processors", *IEEE VLSI Test Symposium*, 2000, pp. 255-262

[4] P. Parvathala, K. Maneparambil, W. Lindsay, "FRITS — a Microprocessor Functional Bist Method", *International Test Conference*, 2002, pp. 590-598

[5] F. Corno, G. Cumani, M. Sonza Reorda, G. Squillero, "Fully Automatic Test Program Generation for Microprocessor Cores", *IEEE Design, Automation and Test in Europe*, 2003, pp. 1006-1011

[6] G. Squillero, "MicroGP — An Evolutionary Assembly Program Generator", to appear on: *Genetic Programming and Evolvable Machines*, 2005

[7] Jin-Hua Hong, Shih-Arn Hwang, Cheng-Wen Wu, "An FPGA-based hardware emulator for fast fault emulation", IEEE 39th Midwest Symposium on, Volume: 1, 18-21 Aug. 1996, pp. 345-348 val.1

[8] SPARC International, *The SPARC Architecture Manual*

[9] N. Kranitis, G. Xenoulis, A. Paschalis, D. Gizopoulos, Y. Zorian, "Application and Analysis of RT-Level Software-Based Self-Testing for Embedded Processor Cores", *International Test Conference*, 2003, pp. 431-440