POSTER

Evolutionary failing-test generation for modern microprocessors

ERNESTO ERNESTO SANCHEZ SÁNCHEZ, Polytechnic of Turin, Turin, TO, Italy

GIOVANNI SQUILLERO, Polytechnic of Turin, Turin, TO, Italy

ALBERTO PAOLO TONDA, Polytechnic of Turin, Turin, TO, Italy

Published: 12 July 2011

Citation in BibTeX format

GECCO '11: Genetic and Evolutionary Computation Conference
July 12 - 16, 2011
Dublin, Ireland

Conference Sponsors:
SIGEVO

Evolutionary Failing-Test Generation for Modern Microprocessors

Ernesto Sanchez, Giovanni Squillero, Alberto Tonda
Politecnico di Torino
Torino, ITALY
{ ernesto.sanchez, giovanni.squillero, alberto.tonda } @polito.it

ABSTRACT

The incessant progress in manufacturing technology is posing new challenges to microprocessor designers. Nowadays, comprehensive verification of a chip can only be performed after tape-out, when the first silicon prototypes are available. Several activities that were originally supposed to be part of the pre-silicon design phase are migrating to this post-silicon time as well. The short paper describes a post-silicon methodology that can be exploited to devise functional failing tests. Such tests are essential to analyze and debug speed paths during verification, speed-stepping, and other critical activities. The proposed methodology is based on the Genetic Programming paradigm, and exploits a versatile toolkit named µGP. The paper demonstrates that an evolutionary algorithm can successfully tackle a significant and still open industrial problem. Moreover, it shows how to take into account complex hardware characteristics and architectural details of such complex devices.

Categories and Subject Descriptors
B.7.2 [Integrated Circuits]: Design Aids – *Verification*

General Terms
Algorithms, Experimentation, Verification.

Keywords
Microprocessor, Post-silicon, Failing-test

1. INTRODUCTION

Nowadays, manufacturing technology is advancing at a faster pace than designing capability, posing unprecedented challenges in the arena of integrated circuits. The so-called *verification gap* denotes the inability to fully verify the correctness of devices that could be built, and indeed *are* actually built. Practice surpasses theory: comprehensive verification of a chip can only be performed after tape-out. Once manufacturing is completed and first silicon is produced, the early chips are sent back to their design teams. This process is called *post-silicon verification* to distinguish it from the traditional, pre-silicon, one. More generally, several activities that were originally supposed to be part of the pre-silicon design phase are nowadays migrating to the post-silicon time. The cost of manufacturing prototypical devices is enormous, but this practice is not an option. Designers candidly acknowledge that "very few chips ever designed function or meet their performance goal the first time" [1].

Microprocessors are a paradigmatic example of the current trend: devices for the desktop market contain billions of transistors, implement complex architectures, and operate into the microwave frequency range. The design flow of a modern microprocessor goes through several iterations of frequency pushes prior to final volume production. Such a process is called *speed stepping*. A *speed path* (or *speedpath*) is a path that limits the performance of a chip because a faster clock would cause an incorrect behavior. Speed paths may be the location where potential design fixes should be applied, and may indicate places where potential holes in the design methodologies exist.

At design time, the slowest logic path in a circuit is termed the *critical path*, and it can be quite easily determined. However, for complex high-performance designs it has been recognized that critical paths reported from the pre-silicon timing analysis tools rarely correlate well to the actual speed paths. The reason is that any pre-silicon analysis tool is only as accurate as the model and the algorithms it uses. Obtaining 100% accurate process models for nanometer processes is difficult, if not nearly impossible. Analysis algorithms are also approximated because of the complexity involved. Moreover, timing behavior on the silicon is a result of several factors mingled together. But in the pre-silicon phase it would not be computationally feasible to consider all these factors simultaneously, and they are analyzed separately. The identification of *failing tests*, i.e., sequences of operations that uncover incorrect behaviors when run at high frequency, is highly related with speed path identification.

The identification of *failing tests*, i.e., sequences of operations that uncover incorrect behaviors when run at high frequency, is highly related with speed path identification. Failing tests may be, for example, sequence of inputs to be applied to the microprocessor pins by an automatic test equipment (ATE). Such test are usually crafted with care by engineers starting from the pre-silicon verification test suite; generated by pre-silicon specialized tools, or automatic test pattern generators (ATPGs); or also created on silicon[1], tackling the actual devices [2] [3]. This work shows how effective failing can be automatically generated *on silicon* by an evolutionary algorithm.

2. GENERATION AND EVALUATION OF TEST PROGRAMS

The toolkit exploited in this work is called µGP (MicroGP) [4], available under the *GNU Public License* from *Sourceforge*[2]. Unlike usual genetic programming (GP) implementations, µGP specific target is to produce realistic assembly-language programs. Its original purpose was to assist designers in the generation of programs for the test and verification of different microprocessors, hence, the Greek letter micro in its name.

For the core can be simply used out of the box, the efficacy of an evolutionary algorithms depend on other factors. The two most important one are: what feedback is used to evaluate candidate solutions, termed *fitness* by the evolutionary algorithm scholars; what is encoded inside individuals. While exploiting an evolutionary

GECCO'11, July 12–16, 2011, Dublin, Ireland.

[1] The expressions "post silicon" and "silicon based" are also used.
[2] http://sourceforge.net/projects/ugp3/

approach is *per-se* of little interest, selecting and tuning such elements can effectively enable to find a solution.

Roughly speaking, desktop microprocessors are made using the complementary metal-oxide-semiconductor (CMOS) technology, based on field-effect transistors (FETs). In such devices, reducing the voltage increases the time required to switch between logic values. Increasing frequency and reducing voltage involves significantly different phenomena in the physical world, especially where not all paths have the same Vcc sensitivity or where paths are interconnect dominated. However, the effects of reducing voltage may be related to the effects of increasing the operating frequency. Thus, for the sake of a feasibility study, the evaluation of a candidate test could be based on its functional core voltage, and on the percentage of runs that actually failed. It must be noted that there is no *conceptual difference* between overclocking and undervolting from the point of view of the proposed algorithm.

Candidate failing test include a mechanism that helps checking their own correctness: all the results of the calculations performed by the test program are compacted in a single signature using a hash function. The evaluator first runs the test program in safe conditions and store the signature. Then it runs the program again at decreasing CPU core voltages, checking that the signature is not modified. As soon as a difference is detected, the functional voltage threshold is recorded. The whole process is repeated R times to tackle variability.

The internal representation is another key aspect. The evolutionary algorithm must be given the opportunity to generate useful solutions. For the generation of failing test is performed during the speed stepping or incoming inspection, it is essential to test all possible instructions, and especially the newest. The assembly instructions made available to µGP can be divided in three main classes: *Integer instructions*, *x87 instructions*, *single-instruction/multiple-data* (SIMD). Not surprisingly, SIMD instructions are particularly critical during speed stepping. The complex calculations involved by these instructions cause data to go through several functional units, and the resulting *datapaths* are prone to be source of problems when the operating frequency is increased.

Cache memories must be taken into account as well, since there may be a significant difference in performance and power consumption between a L1 cache hit and a L1 cache miss. In order to give µGP the possibility to generate cache hits and cache misses, a special set of C variables was defined. The variables are carefully spaced so that all their memory locations will be cached in the very same cache location. If the microprocessor uses a k-way set associative L1 cache and $C > k$, a shrewd sequence of read and write operations on such variables may generate the desired cache activity. It must be noted that the goal of adding such variables is to let the evolutionary core to control the cache activity, but no suggestions are given on how to exploit them. µGP would find autonomously which sequence of operations is more useful to generate a failing test.

3. EXPERIMENTAL EVALUATION

While no working methodology for functional failing-test generation has been reported in the specialized literature yet, a related problem is faced by a community of computer enthusiasts. *Overclockers* try to push the performance by increasing the operating frequencies of their microprocessors and the CPU core voltages [5]. However, after pushing their computers to astonishing frequencies, they need to assess the stability of their systems. The test suites that are used to stress the systems and highlight criticalities may be regarded as generic fail tests not focused on a specific microprocessor. Thus, they can be used as a baseline to evaluate the performances of the proposed methodology.

Most of the information about stability stress tests is available through forums and web sites on the internet, with few or none official sources. However, there is quite a generalized agreement in the overclockers community on these tools: *SuperPI* and *CPU BurnIn* are rather old, but have been included for the sake of comparison. *Prime95* has become extremely popular among overclockers as a stability test. *LINPACK* is a software library for performing numerical linear algebra on digital computers. *LinX*, *IntelBurnTest*, and *OCCT* exploit *LINPACK* to assess the stability.

Experiments were run on an *Intel Pentium Core 2 Duo E2180*, sll experiments have been repeated 10 times. µGP parameters are shown in Table I. Table II compares the proposed methodology with older stress tests.

TABLE I. **FAILING-TEST REQUIRED TIME FOR SINGLE THREAD**

CORE V	SuperPI	CPU BurnIn	µGP
1.2625	7″	5′	≤ 1″
1.2750	10′	> 10′	≤ 1″
1.2875	> 10′	> 10′	> 10′

Failing tests devised with the proposed methodology clearly outperform all the other approaches. Remarkably, µGP was asked to find a very fast failing test for a specific microprocessor, and therefore there is no guarantee that the devised program would fail on a different model. Moreover, the test was required to be very short, to avoid heating effects. On the contrary, stress tests intentionally exploit overheating and are designed to work with different architectures.

TABLE II. **FAILING-TEST REQUIRED TIME FOR MULTIPLE THREADS**

CORE V	Prime95	IntelBurnTest	LinX	OCCT	µGP
1.2625	1″	2′	2′	3″	≤ 1″
1.2750	6″	2′	2′	4″	2″
1.2875	4′	4′	2′	7′	2″
1.3000	> 10′	7′	7′	> 10′	10″
1.3125	> 10′	> 10′	> 10′	> 10′	8′
1.3250	> 10′	> 10′	> 10′	> 10′	> 10′

The final failing test is 614 line long. The two functions executed by the two cores are respectively 280 and 235 line long. The remaining lines are mainly used to define and initialize variables or other program parts. µGP requires 5h for generating the failing test.

It must be also noted that the temperature of the microprocessor during the experiments never exceeded 40°C, while running LINPACK-based stress tests it is permanently above 45°C.

4. REFERENCES

[1] R. McLaughlin, S. Venkataraman, and C. Lim, "Automated Debug of Speed Path Failures Using Functional Tests," in *The 27th IEEE VLSI Test Symposium*, 2009, pp. 91-96.

[2] L. Lee, L.-C. Wang, P. Parvathala, and T. M. Mak, "On Silicon-Based Speed Path Identification," in *Proceedings of the 23rd IEEE VLSI Test Symposium*, 2005, pp. 35-41.

[3] J. Zeng, J. Wang, C.-Y. Chen, M. Mateja, and L. .-C. Wang, "On evaluating speed path detection of structural tests," in *11th International Symposium on Quality Electronic Design (ISQED)*, 2010, pp. 570-576.

[4] E. Sanchez, M. Schillaci, and G. Squillero, *Evolutionary Optimization: The µGP Toolkit*. Springer, 2011, ISBN: 978-0-387-09425-0.

[5] B. Colwell, "The Zen of overclocking," *Computer*, vol. 37, no. 3, pp. 9-12, 2004.