PHYSICAL FAULT INJECTION: A SUITABLE METHOD FOR THE EVALUATION OF FUNCTIONAL TEST EFFICIENCY

R. VELAZCO, B. MARTINET
L.G.I.- IMAG
46, Av. Félix Viallet
38031 - Grenoble

Abstract

As a matter of fact the last generation of 32-bits microprocessors seems "to ring the knell" of functional test methods. Indeed, those based on fault hypothesis cannot cope with the technological reality; the other ones (ad hoc tests, systematic test,...) are overwhelmed by the very large number of cases (instructions) to be analyzed.

In this paper we present a pragmatic approach that attempts to resolve this critical problem: physicals faults are injected by means of a microulting laser equipment on a set of good circuits in order to evaluate the efficiency of different test sequences. This approach is illustrated by an actual experiment performed on more than an hundred of 68000 microprocessors.

Introduction

Since 1978 functional test methods have appeared as an economic solution, from a user point of view, to the problem of test generation for complex circuits. These methods, in opposition to structural test methods, are only based on user available informations (instead of informations concerning the actual circuit implementation). Most investigations realized in this area constituted attempts to extend test generation techniques used at the logic level to the functional level: both from a formal model of the circuit to be tested and a set of hypotheses about how the physical faults will manifest at the considered model level, test sequences detecting a subset of the postulated faults are derived. On the one hand, a considerable research effort has concerned the minimization of the test sequence length [1-3], on the other hand, few results have been published about the efficiency, when applied to actual circuits, of the proposed test sequences. The evaluation method more frequently evoked, but unfortunately seldom applied, was the injection of stuck-at faults in an equivalent logic model of the circuit to be tested [4].

Ideally, a "good" functional test program is a program that exercises with appropriate data all the functionalities of the target circuit. For a given microprocessor, this entails exercising at least once: each of the valid instructions, i.e. each of the valid combinations of opcodes and addressing modes, and each of the representative configurations of asynchronous signals (interrupts, reset, halt,...). Obviously, the enormous possibilities of most microprocessor instructions sets avoid the manual generation of such a test program, making then difficult its application by means of usual commercially available testers.

In order to face these "practical" problems, we have designed and developed both a software tool - the GAPT test program generator - and a range of experimental functional testers - the TEMAC and FUTE16 prototypes [5,6]. From a high-level description of the target instruction set, the GAPT software generates a well structured test program that completely activates the whole set of considered addressing modes and opcodes combinations; each one of the tested instructions being surrounded by the initialization of the corresponding source registers and the observation of the destination registers.
The test programs are generated in the assembly language of the tested microprocessor, the object code is then downloaded to the tester memory without any transformation. In the testers we have realized, the DUT itself controls the program flow, fetching instructions and data from the tester memory and writing results in the same memory. Our testers provide then a "natural" microprocessor environment leading to a significant test code compaction (the reduction ratio depends on the average number of cycles per instruction of the considered microprocessor), and a better understanding of test results (improving then the diagnosis capabilities). The major drawback of this kind of test system concerns its limitations in either the type of tests (only logic tests) and the type of target circuits (circuits able to execute a program).

Purpose

The project presented in this paper is an extension of the GAFT project and focuses with the automatic test program generation for complex processors such as 32-bits present CISC and RISC microprocessors. The analysis of the instruction set of these circuits shows that their exhaustive exploration (basic principle of the GAFT method) is unpractical. The main difficulty comes from both the large number of general purpose registers and the complex addressing modes involving frequently three parameters (two registers for the addressing mode and a scale), leading in some cases to hundred thousands of valid configurations. For example, a complete development of the address register indirect with index (base displacement) addressing mode of the 68020 microprocessor (displacement size*base register*index*scale) leads to 3487 cases. If the registers (base and index) and the scale are considered as parameters, the number of cases is reduced to 18. Moreover, the verification of all ADD instructions of this microprocessor needs about 13x10^6 instructions, whereas in the case of parameter limitation, this number is 2100. The question is: what's the efficiency of such a reduced functional test program? Even if it is obviously impossible to give a general answer to this question, as it involves the actual circuit implementation, we attempt in this paper to give initial elements showing the possibilities that are offered by this variant of the GAFT method.

Thus, in order to cope with the complexity of present 32-bits instruction sets, a new functional test generation mode, the so-called "reduced conformity test", has been added to the GAFT software. It consists in defining the parameters sets in which a random drawing, instead of an exhaustive exploration, will be performed during the test program generation. The choice of such sets must take into account informations (or well founded suppositions) about the actual implementation of the control part. Typical examples of this parameters sets are the index register, the address register, etc.... Indeed, in most microprogrammed controllers, the treatment is independent of the register numbers.

As an example, the functional test program of the MC68000 16-bits microprocessor, used to measure the efficiency of the functional test [7], which required 60 test modules of 60 Kbytes, becomes a single module (with less than 60 Kbytes) test program when the "reduced conformity" test mode is used.

Experimental results

Description

For a "stable" circuit as the 68000 microprocessor, the escape rate of manufacturing testing (defective circuits declared good) could be a few tens of parts per million. These circuits are interesting because they have probably a sharp single defect, but they are unfortunately seldom available. So, it has been decided to test circuits for which single defects, with a random distribution on the chip surface, were artificially created. As in a previous experiment [7], physical...
faults have been injected by means of a laser, allowing to cut either aluminium or Sipoly tracks.
The beam position accuracy was 0.1 μm for X and Y coordinates. For a couple of X and Y coordinates randomly generated, the nearest Alu or Sipoly track was looked for and cut (except power supply track).
A sample of 140 defective microprocessors (68000) was then set up. Figure 1 shows the distribution of the created faults on the chip surface.

Results

The main goal of our experiment is to compare the efficiency of two test programs generated by the GAPT method: a complete functional test—the so-called conformity test—and a reduced functional test—the so-called reduced conformity test. The manufacturer test is used as a reference test.

Among the 140 "defective" circuits, 12 are declared good by all the tests. This fact can be explained in two ways. Either the injected defect has no consequences in the circuit behavior (redundancy, useless track,...) or the cut was incompletely performed (provoking only a track shrink).

The remaining 128 circuits are found faulty by the three test programs. Nevertheless, for 11 of this circuits, the reduced conformity test sometimes declare them good. Then, if this circuits were tested only once, they could be declared good. The detection of the created defect seems
require some conditions that are obtained with a repeated execution of the test program (temperature, charge, leakage current . . .).

As shown in the figure 1, stating a correlation between the area concerned by the defect and the nature of the induced faulty behavior seems to be an impossible task if the implementation details are unknown.

Concluding remarks

We have presented the result of an experimental method that allows to evaluate the efficiency of functional test programs, based on physical fault injection. This experiment shows that the length of functional test programs can be drastically reduced without significant decrease of fault coverage. Nevertheless, the 100% detection was attained only when the reduced conformity test program was looped. This could mean that the test program length has an influence (probably indirect) in the detection of some faults. If we consider present 32 bits processors, their high complexity will lead to a reduced conformity test program having an important length which could mitigate this problem.

Obviously, these are only preliminary results that need new experiments to be confirmed. In particular, the type of the created defects must be enlarged to represent a wider set of actual physical faults.

New experiments including shorts creation between Alu tracks (bridge faults) are now in progress.

Another possible orientation is the random injection of defects in different areas (ALU, registers, PLAs, . . .) of the circuit instead of in the whole chip surface. The global coverage of a given functional test program can be evaluated as a weighted sum of local coverages, the weights being derived from features of each area such as number of transistors, number of tracks, probability of each type of faults (open, short, . . .).

Acknowledgments

The authors wish to thank G. Auvert, French Center of Telecommunication (CNET-CNS) for helping to carry out the laser experiments. We also like to express our thanks to J. Beausse & J. Feltrin, Thomson TMS, for providing and testing 68000 microprocessors.

References