DE GRUYTER OPEN

Journal of ELECTRICAL ENGINEERING, VOL. 66, NO. 4, 2015, 185–193

AN EFFICIENT FUNCTIONAL TEST GENERATION METHOD FOR PROCESSORS USING GENETIC ALGORITHMS

Ján Hudec — Elena Gramatová [*]

The paper presents a new functional test generation method for processors testing based on genetic algorithms and evolutionary strategies. The tests are generated over an instruction set architecture and a processor description. Such functional tests belong to the software-oriented testing. Quality of the tests is evaluated by code coverage of the processor description using simulation. The presented test generation method uses VHDL models of processors and the professional simulator ModelSim. The rules, parameters and fitness functions were defined for various genetic algorithms used in automatic test generation. Functionality and effectiveness were evaluated using the RISC type processor DP32.

K e y w o r d s: processor, testing, functional test, test generation, genetic algorithm, evolutionary strategy

1 INTRODUCTION

New technologies, higher chip complexity and increasing clock frequencies give new challenges and issues for testing complex digital systems integrated on a chip (SoC — system on chip). Testing and design for testability of SoCs play a prominent role not only in design and manufacture processes but also during their lifetime in applications. Processors are basic and the most complex elements in SoCs therefore their testing needs continually new test generation methods and test application techniques during their verification, SoCs manufacture testing and reliable lifetime run. Traditional test techniques based on sequential and structural automatic test pattern generation (ATPG) are too computational demanding for circuits owning the complexity of today's processors [1]. In this context, functional testing processors achieves "come back" in research and practical cases targeted mainly to development of the new functional test generation methods which can be efficiently automatized together with test quality evaluation.

Recently, new approaches and the test generation methods have been developed and published which are based on self-testing and instruction set architecture (ISA). The test generation methods, named software-based self-testing (SBST) methods, have been developed for processor models at register transfer level (RTL). Self-testing means that the tests can be performed on a tested processor without using an automatic test equipment (ATE) and/or as additional tests to structural tests of processors integrated in SoCs after their manufacturing. Such self-tests can be used also during verification in a design process of processors and SoCs with processors. In addition, the functional tests can be applied any time when the processor is in idle time during its or SoC lifetime if necessary. Software-based testing means that the functional tests for a processor consists of various ordered sequences of instructions from its ISA and selected data memorized in registers. Such methods are especially attractive when no structural information about processors is available and they are independent on used technology. It is their advantage that can be reused for the processors implemented in another technology or in various SoC applications. Minimum or even no circuit modifications are needed with SBST to run the test and it is performed in normal operating mode. Moreover, once the test code has been uploaded in the system, the test is completely autonomous, and can be run at speed relying on a free running clock provided *eg* by used ATE. Besides of fault coverage other important aspects are needful to consider as the code length, the test program run duration and power consumption [1].

Different approaches and methods can be found in literature for the functional test generation, addressing various fault models and based on assorted techniques. Several methods have been published for the SBST tests generation [1–5] and some of them use genetic algorithms (GAs) [6–9]. Quality of generated tests is solved over the processor model using a hardware description language (HDL), *eg* VHDL, Verilog and professional simulators or fault simulators based on specified fault models. The simplest ATPG approach is to use the professional simulators with functionality of covered codes calculation, covered registers during data transfer over the processor model. It was experimentally evaluated that if a SBST test shows high code coverage then also can achieve high and sufficient coverage of stuck-at faults (stuck-at 0, stuck-at 1). The high code coverage means app. 90 % code coverage [8].

[*] Faculty of Informatics and Information Technologies of the Slovak University of Technology Ilkovičova 2, 842 16 Bratislava, Slovakia, jan.hudec@stuba.sk, elena.gramatova@stuba.sk

DOI: 10.2478/jee-2015-0031, Print ISSN 1335-3632, On-line ISSN 1339-309X © 2015 FEI STU

The functional test consisting finite ordered sequences of instructions is named as test program [9] or test mix [10] (this term is used in the paper). Various methods for automatic test mixes generation have been published till now and the most of them are based on GAs with one selected evolutionary strategy (ES) and randomly selected the first population [6, 8]. Based on these results, a new test generation method has been developed using different types of GAs with specific first population and possibilities to change ES according feedback quality of a generated test mix. The main features of the new test generation method are adaptability, flexibility and lower time consummation for finding optimal test mixes with accepted code coverage. Some rules and strategies have been defined for the new method and its automatic implementation for developing an effective ATPG for processors. The new method has been constructed for VHDL (very high speed integrated circuits description language) models of processors and their ISA. A software environment has been designed based on using simulator ModelSim for code coverage calculation [11]. Experiments shown efficiency, flexibility and adaptability of the developed test generation method with very good code coverage results and short time consuming for test mixes generation in comparison with published results.

The paper is organized as follows. The next section presents a state of the art about functional testing processors based on SBST principles. Section 3 describes the new test generation method, defined rules and strategies for using GAs with feedbacks and changing ESs, suitable for automatic implementation. Effectiveness of automatically generated test mixes has been evaluated by the code coverage in experiments over a processor model in VHDL and using the simulator ModelSim. Experimental results over the DP32 processor model in VHDL and its ISA are presented in Section 4 before conclusion.

2 RELATED WORK TO SBST METHODS

The key idea of SBST type of functional tests for processors is to exploit on-chip programmable resources to run normal programs that testing a processor itself. The processor applies functional test programs — test mixes using its native ISA with elimination of the need for an additional test-specific hardware. Such test mixes are applied at the processors actual operating frequency that is advantage in comparison with ATE structural testing.

A test mix (test program) is a valid sequence of finite and ordered assembly instructions, that is fed to the processor through its normal execution instruction mechanism (*ie* the processor executes it like any other functional program). The main goal of the test program is to cover any possible design or production defects in the developed or integrated processor in SoCs or embedded systems.

The SBST features and advantages are [1]:

- *Non-intrusiveness.* SBST does not need any processor modification and no extra power consumption is produced in comparison to the normal operation mode.
- *At-speed testing.* Test program application and responses collection are performed at the processors actual speed, which enables screening of delay faults that are not observable at lower testing frequencies.
- *No over-testing.* SBST concentrates on the same circuitry used in the normal processor operations and therefore avoids test overkill, and thus detection of faults that would never manifest during the normal processor operation. This leads to significant yield gains and also shorting test time.
- *In-field testing.* Self-test programs from manufacturing testing can be reused in the field throughout product lifetime which is very important in nowadays complex applications.

Unfortunately, the SBST methods show some limitations and disadvantages [4]:

- Some faults can modified the test program flow and potentially can leaded to an endless execution, making it difficult to take back the control of the system at the end of the test.
- Some memory addresses are never accessible because not reserved to the test procedure, therefore resulting in a coverage loss.
- Size and execution time of the test could be prohibitive in case of stringent real-time application requirement.
- IP (intellectual property) protection is not guaranteed, since the test program may reveal details about the processor core implementation.

Designers and test engineers have to decide where and when the SBST test can be applied for processors based on the mentioned advantages and disadvantages. Currently published SBST approaches and methods do not necessarily aim to substitute other established functional or structural testing approaches but rather to supplement them by increasing test quality at low cost. Evaluation of generated test programs can be done at the functional level using coverage metric over a processor model using a professional simulator or a fault simulator for stuck-at faults coverage at the structural level. The principal scheme for SBST test programs generation using the structural fault simulator is shown in Fig. 1. A test program is created from an instruction set library and its quality is evaluated by fault simulation. If fault coverage is low thus a new test program is generated. The test program generation is ended if satisfactory coverage is achieved for all test programs involved in a test for the tested processor. The fault simulator can be replaced by a logic simulator when only a code coverage metric is calculated. The code coverage metric is defined as coverage of statements of used HDL description, or registers in RTL processor description, branches, toggle dates *etc.* The test programs generation also finish if code coverage

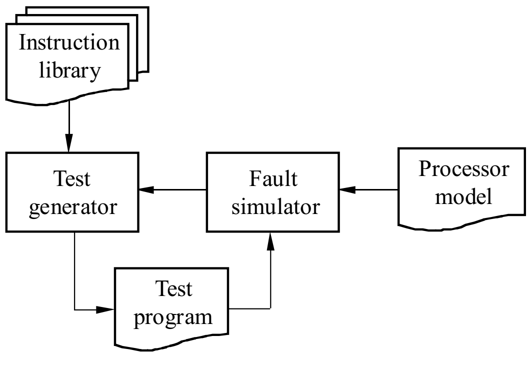

Fig. 1. Principal scheme for test programs generation

is satisfactory. Both types of test programs evaluation are used in the published SBST test generation methods.

Achieving the test quality target requires a proper test program generation phase which is the main focus of the most SBST methods. Due to many test generation approaches and methods published in the literature, the review of the previous works is mainly concentrated to the methods based on GAs because they are turned out to be the most effective in automatic realization. GAs are optimization algorithms based on natural genetics and selection mechanisms [1]. As an adaptive search technique, GAs have been used to find solutions to many NP-complete problems and have been applied in many areas as well as in test generation at the functional level. Each of GAs uses a set of operators (selection-reproduction, crossover, mutation) and an objective fitness function. The fitness function is used for evaluation of each individual population before application of GA operators for creation of a new population [4, 7, 9]. The fitness definition depends on selected and used coverage metric in SBST test methods based on GA.

One of the first automatized SBST test generation method [12] is based on a system graph where nodes represent registers and arcs showing data transfer and they are evaluated by mnemo-code of instructions using the node registers [13]. The method was applied to a RISC (reduced instruction set computer) processor based on knowledge of its ISA and defined functional fault model. Evaluation of the proposed method was done over the processor core DP32 (the 32-bit RISC type processor) using an in-house developed structural fault simulator and stuck-at fault coverage achieved 96 %.

The other two test generation methods exploit a combination of processor abstraction models, such as RTL descriptions and ISA specifications. An ATPG environment has been developed consisting of a structural fault simulator and various deterministic test generation algorithms applied for the different modules of the processor. The resulting test programs were loop-based pieces of code that deterministically provide the modules under testing with a series of data inputs carefully selected. The methods were based on analysis of data dependencies of available SBST programs and general parameters of the pipeline architecture and memory system with remarkable results — 95 % stuck-at fault coverage for two RISC processors with pipelined architectures: MiniMIPS and OpenRISC 1200 [14].

Similar ideas are used in [15] for generating SBST tests for multiprocessor cores integrated on a chip. The deterministic method was optimized for SBST tests generation on multiprocessor Sun OpenSPARC T1. The high level, the component-oriented SBST method which achieved a high stuck-at-fault coverage for an embedded RISC Plasma/MIPS processor core was presented in [3]. The method is based also on the deterministic algorithm with knowledge of ISA and RTL description of the tested processor. Easy detectable faults were detected by random test generation and hard detectable faults were detected by D-algorithm for all functional components. The total obtained fault coverage was the highest fault coverage among all works applied to Plasma/MIPS processor core — 96 %.

The next methods for automatic test programs generation start with application of a selected GA and evolutionary strategies where feedback information from a linked simulator can have influence to creation of a new population [6–9, 16]. New system and tool based on evolutionary strategies [6–9], named μGP [16], is presented and used for automatic test programs generation. This evolutionary tool is composed of test generator, fault simulator, instruction library and processor model modules. The idea is based on the next two actions. The first one is targeted to random generation of a population consisting μ individual test programs. Then they are ordered according the fitness defined by the used code coverage metric. The new individuals for reproduction are achieved by means of tournament selection. Such a way the λ new individuals from the parents μ are created and each individual is formed with configurable number of genetic operators crossover and mutation. The second one is associated with use of the evolutionary strategy $(\mu + \lambda)$ — ES$(\mu + \lambda)$. All new unique individuals are then evaluated, and the population resulting from the union of old and new individuals are sorted by decreasing fitness. Finally, only the first λ individuals are kept. The condition for GA termination is the following: A target fitness value is achieved by the best individual; no fitness increase is registered for the predefined number of generations; a maximum number of generations is reached.

In [6] the μGP system was used for automatic test programs generation for 32-bit processor Leon2 with SPARC V8 architecture. The ModelSim simulator was employed for simulating the design and calculating the statement coverage. The developed instruction library for the Leon2 contained about 500 syntactical descriptions of the possible instructions and their operands. The adopted validation metric was the RTL-instantiated (instantiated means the calculation of the metric against the elaborated design, rather than on source description) statement coverage: the percentage of executed RTL statements to the total statements number in simulating of a given test program execution. The used fitness function was the direct measure of the test programs attained coverage. As a result, μGP devised a test program set attaining 100 % statement coverage (experiments used neither floating point unit nor coprocessor). Also the effect of targeting different coverage metrics with respect to validation and test was evaluated.

Another case study and results with GP have been published in [7] and improved in [9]. The usability and effectiveness of μGP were tested by the Intel i8051 processor core. The experiments showed fault coverage up to 93.6 % where an in-house developed parallel fault simulator FENICE was used as the external evaluator linked to the μGP tool [16].

The case study with use of μGP and the DLX/pII processor core is described in [8]; the professional simulator ModelSim [11] was used as an external evaluator. The code coverage with different metrics was used instead of fault coverage. The goal of the experimental evaluation was twofold: effectiveness validation of the test programs completion and optimization with different coverage metrics with respect to the test programs generation. All experimental results reported on this processor core were focused on reusing already developed test programs and show effectiveness of the selected approach. The attained results were all saturated high above 90 % in statement and branch coverage and above 80 % in condition and toggle coverage. The use of the proposed approach enabled to experimentally analyze the relationship of the different code coverage metrics used in test program generation and code coverage figures. It can be used as a heuristic indication for guiding the test generation suitable also for processors verification.

All SBST methods based on GAs generate test programs (test mixes) by one type of GAs and one used evolutionary strategy ES$(\mu+\lambda)$. They start with a random test mix as a started population in GA. Some of them use feedbacks about quality of a generated test mix using a processor simulator that is able to elaborate with the test and returning code coverage or stuck-at fault coverage in using the structural fault simulator. Obviously the main drawback regarding these test generation methods is the computational effort involved to generate a good test mix in each population. The idea to create the first population as best as possible and to use changing ES based on feedback results have been motivation to develop a new SBST generation method using different operators and ES in GA application for finding an optimal set of test programs (test mixes) with sufficient coverage for specified and used coverage metric at RTL. The new SBST method is described in the next section and the experimental results are presented over the DP32 processor core [17] described in VHDL.

3 A NEW ADAPTIVE SBST METHOD

A new SBST method has been developed based on mentioned aspects at the end of Section 2. The new test generation method uses GAs together with changing various ES and VHDL processor models. Alternation of evolutionary strategies is based on feedbacks from coverage evaluation and specified parameters p and k (defined in part 3.1). The code coverage is used as the coverage metric over the VHDL processor models in dependence on a linked digital simulator. Simulator ModelSim is considered in the method and experiments are presented in section 4. Notation "test mix" is used in the developed method instead of "test program" as a SBST test for a processor. This term has been defined concurrently [10] with other publications where "test program" was defined with the same meaning. Terms test mix and the coverage metric are defined in part 3.1 of this section.

3.1 Basic definitions and notation

Notation and terms used in the developed SBST test generation method described in part 3.2 are defined below.

Definition 1. A test mix is valid dependent and finite sequence of instructions from ISA of a tested processor and defined operands.

Definition 2. A SBST test is a valid meaningful sequence of generated test mixes.

Definition 3. Diagnostic coverage of a test mix is code coverage of a processor model in VHDL

Definition 4. Code coverage is quality validation of a test mix based on criteria: statement coverage, branch coverage, condition coverage, toggle coverage.

Definitions and characterization of the code coverage criteria are described in [8, 11]. Fitness function is defined as a numeric value representing the code coverage.

Definition 5. Fitness function (fitness) for one test mix M_j is defined by expression $F_j = w_s s_j + w_b b_j + w_c c_j + w_t t_j$ where $j = 1, 2, \dots, N$, N is the number of test mixes, then w_s, w_b, w_c and w_t are the corresponding predefined weights of statement, branch, condition, toggle coverages and s_j, b_j, c_j, t_j are their coverages.

Definition 6. The weights w_s, w_b, w_c, w_t are values from interval $\langle 0, 1\rangle$ and they have to satisfy condition $w_s + w_b + w_c + w_t = 1$.

Two ES are involved in the method: ES$(\mu+\lambda)$ described in the previous section and ES(μ,λ). The main difference of the second one is selection of new λ individuals for the next generated population where the old μ parents are not involved to the next population [18]. Therefore it should be useful to use both strategies and also more parameters as lifetime of each population (k) or probability of mutation (p). Using both strategies with the mentioned parameters in the SBST method can generate high quality test mixes in shorter time. They also increase adaptability and flexibility of the SBST method. Strategy $\mathrm{ES}(\mu, k, \lambda, p)$ used in the developed SBST method is the advanced and relatively latest ES defined in [18]. It hasn't been used in GAs practical applications and not in the existed SBST methods till now.

Definition 7. Evolutionary strategy $\mathrm{ES}(\mu, k, \lambda, p)$ is based on pseudorandom selection of r individuals ($r \geq \lambda \geq \mu$) from μ parents, executing of operations by genetic operators, ordering the obtained individuals with decreased fitness and selection of λ individuals to the new generated population with the best fitness in accordance of selected value k $(k \geq 1)$.

Definition 8. Consider strategy $\mathrm{ES}(\mu, k, \lambda, p)$ and a) if $k = 1$, then strategy $\mathrm{ES}(\mu, \lambda)$ is adjusted, b) if $k = \infty$, then strategy $\mathrm{ES}(\mu + \lambda)$ is used.

Definition 9. Diversity is a quantitative measure characterized by existing population of individuals (test mixes) and indicates how many different types of test mixes are involved in population.

The diversity value is dependent on the number of individuals in the population and the evenness of individuals.

Definition 10. Evenness is a quantitative measure of type representations in existing population, which quantifies how equal is the population of testing mixes numerically.

The Shannon-Wiener index and/or standard deviation [18] have been originally proposed and used for diversity and evenness computation of fitness.

Based on the previous definitions some parameters have to be specified and adjusted in the SBST test generation method. The major of them are:

- Number of generated populations (test mixes) or termination of test mixes generation based on defined diagnostic coverage.
- GA parameters: ES alternation, used operators, starting population.
- Weights and code coverage specification for fitness function.

The other parameters of the SBST method concern of test mixes characteristics and requirements. The major of them are:

- Test mix length depending of ISA complexity.
- Dependability of instructions sequence in a test mix.
- Operands in instructions, if needed have to be defined randomly or targeted to faults in registers and data transfers.

Definition 11. Test mix length is the number of instructions involved in one test mix.

The order of instructions involved in a test mix has to run in the specified order because its changing creates a new test mix. Therefore the order has to be unique for each test mix.

Both types of mentioned parameters have to be specified for SBST method application for each VHDL processor model to receive an optimal test with highest code coverage.

3.2 The SBST method description

The new strategies are defined for the presented SBST generation method using GAs with the goal to increase quality of generated test mixes and to find an optimal test for processors in short time.

Strategy 1. Deterministic or pseudorandom generation of an initial started population (test mixes) with higher code coverage can increase quality of the final functional test consisting several test mixes.

Strategy 2. Application of the advanced ES in which all the basic ESs can be combined with using different methods of genetic operators for selection and combinations of crossover and mutation can improve finding optimal final solution for test mixes and thus for functional testing processors.

Strategy 3. The fitness function designed more sophistically in GA contributes to better feedbacks in GA run.

The strategies are basis of the new adaptive SBST generation method consisting of the next steps:

- Choosing an initial population (test mixes) by a selected test generation method from the existed test mix generation methods.
- Evaluation of each test mix using fitness function specified in Definition 5 with characteristics introduced by Definition 6. Assemble ranking lattice in the order of their ranking.
- Application of the ranking in selection of the parents for the next generation of test mixes.
- Creation of next generated test mixes with adaptation of GA parameters by applying appropriate ES and a set of genetic operators (selection-reproduction, crossover, mutation) or with combination of all of the above.
- Application of the above criteria once again on the new population of generated test mixes.
- Specification of stopping criteria (maximal number of generations in population or demanded quality of the code coverage).

The basic scheme of the new proposed SBST method is shown in Fig. 2. The major and new developed blocks are a) Generation of the initial population, b) Fitness evaluation, c) Parameters adaptation. The fitness evaluation is implemented according Definition 5, used operators are known and therefore only the other two blocks are explained.

Generation of the initial population block is based on selecting the started test mix as best as possible. The specific technique has been developed for creation of a

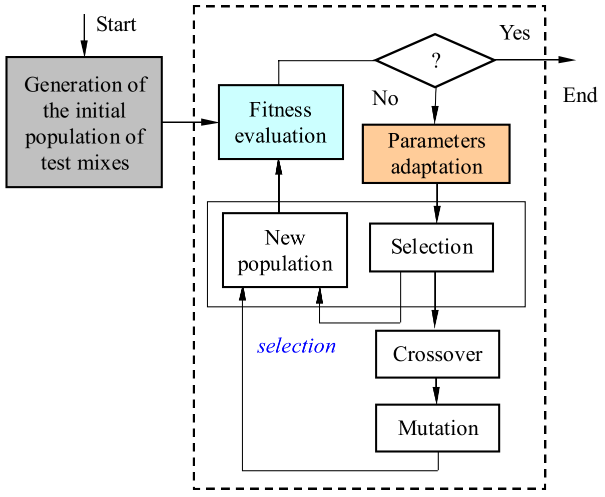

Fig. 2. Basic scheme of the adaptive SBST method

started population based on instruction grouping. The idea is to divide all instructions from ISA of a tested processor into several groups according their characteristics and features, *eg* instruction with one operand, branch instructions, arithmetic operations, logic operations *etc.* Then a test mix is created by adding one instruction from each group according a specified test mix length. Each instruction has the same probability of its occurrence. The other possibilities are to define instruction prioritization or to use the random approach for instruction selection to the started initial population. The best results for generating the started initial population were achieved by the grouping method. The experimental results are demonstrated on processor DP32 and are presented in Section 4.

Parameters adaptation block consists of changing ES, operators in test mixes generation based on diversity and evenness measures during application of one GA. It has been proven [18] that ES($\mu+\lambda$) is very positive in selection of optimal solution and ES(μ,λ) is better in overcoming of local extremes. According Definition 7 loosing parents population can give positive influence to searching progress of new individuals targeted to the best solution.

4 IMPLEMENTATION AND EXPERIMENTAL RESULTS

The proposed and described SBST method was implemented in a new developed complex environment. The implementation environment is based on the architecture shown in Fig. 2, definitions and specifications introduced in the previous sections and final arrangement in more detail is displayed in Fig. 3.

The whole system for realizing the above mentioned ideas is implemented in C# language in environment of Microsoft Visual Studio with using of Microsoft .NET framework on the platform Microsoft windows. The platform provides the opportunity to use whatever programming language for creating the next new modules of the system with respect of universality and flexibility.

All inputs/outputs, except for the evaluating the individuals, are performed using XML format. The use of XML for all input and output operations allows the use of standard tools, such as browsers, for inspection of the constraint and data library, the populations and configuration options. The VHDL language is used for tested processor description and professional simulator ModelSim from Mentor Graphics is integrated to the system

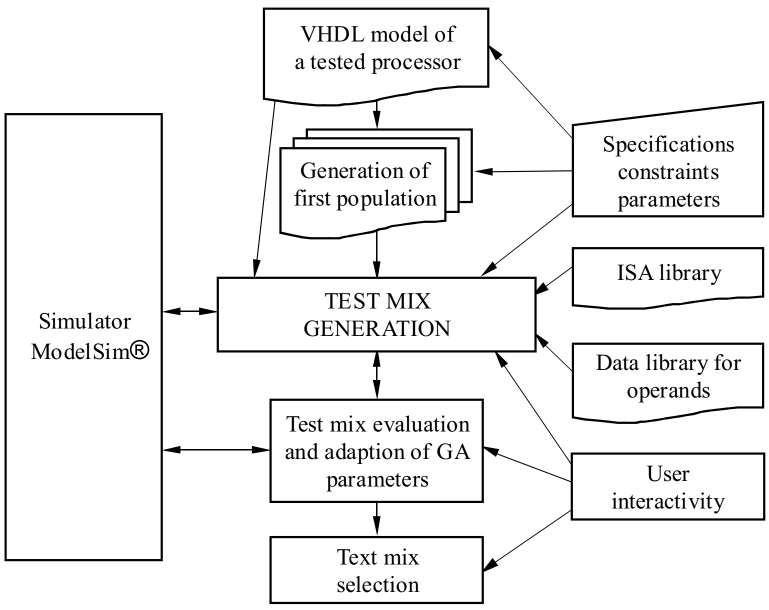
VHDL model of a tested processor
Generation of first population
Specifications constraints parameters
ISA library
Data library for operands
User interactivity
Simulator ModelSim®
TEST MIX GENERATION
Test mix evaluation and adaption of GA parameters
Text mix selection

Fig. 3. Implementation environment for SBST test generation

Table 1. Fitness of initial population with 3 methods for processor DP32

Methods	Test mix length 10 (%)	Test mix length 20 (%)	Test mix length 30 (%)
random	28.6	31.0	38.1
priority	31.5	35.2	49.6
grouping	43.7	52.4	64.3

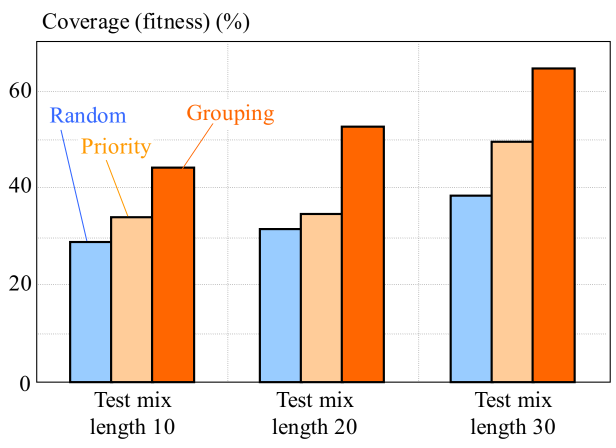

Fig. 4. Experimental results for starting population

for simulation. Block "Test mix generation" is the major block and a hard core of the system with open interface for other extensions. This block realize generation test mixes according ISA library, VHDL processor model and database of operands using specified parameters, alternation of ES described in the previous section. The system is flexible to other extensions, adaptable to various ES with the goal to generate final test with the best test mixes in optimal length and with sufficient VHDL code coverage. Experimental works confirmed expectance of the developed SBST method and they are described in the next paragraph.

In the following the description of the DP32 processor core [17] will be outlined, which is used as a case study in our experiments. DP32 is a RISC type 32-bit processor core with simple instruction set with 20 types of instructions with the length of 32 or 64 bits that are frequently used in programs. It is based on ISA architecture and is a typical representative of ARM (Advanced RISC Machine) processor architecture. The processor core is described as a synthesizable VHDL model for academic and research purposes. Specification and architecture of DP32 is open, portable, non-proprietary and scalable to embedded processors, all sharing the same core (non-privileged) instruction set. It can be implemented in programmable logic such as FPGA or as soft IP cores.

DP32 consists of 32-bit address and date bus, 256 general purpose registers (R0-R255) addressable by software, a program counter (PC), and a condition code register (CC). The memory accessible to the DP32 consists of 32-bit words, addressed by a 32-bit word address. Instructions are all multiples of 32-bit words, and are stored in this memory.

The obtained experimental results of choosing the grouping method for automatic generation of initial population of test mixes are in more details presented in [19, 20].

The values of initial population fitness obtained with using 3 methods of initial population generation (random, priority, grouping) for the DP32 processor core are specified in Table 1.

All simulation code coverages (statement coverage for fitness evaluation was used) were obtained with using each method as average values from 10 experiments, the length of test mix was 10, 20 and 30 instructions. It is visible that the higher values and results with the grouping method achieved as is compared in Fig. 4.

Starting from the initial population of test mixes the fitness function can be calculated for each test mix in this population. The definition of fitness is based on Definition 5 in Section 3.1 and affords the opportunity to select sufficient universal formulation for the specific needs of the tested processor. Then GA with genetic operators (reproduction, mutation, crossover) is performed on the first population of testing mixes, and with reference to the fitness value of each mix. The parents for the next population of test mixes are chosen with changing parameters and stated ES. Some constraints such as elitism of individuals with changing parameter k in $\mathrm{ES}(\mu, k, \lambda, p)$, tournament and roulette selection are also applicable. If the feedback values in the GA progress indicate the small value of diversity or high value of evenness then alternation of parameters in GA have to be adopted. Thus continual adaptation of input parameters k (lifetime of test mix/elitism) and p (probability of mutation) and thus the selection methods (roulette, tournament, ...) are applied until the higher fitness values of generated test mixes are achieved [21]. This procedure is repeating until the best code coverage of test mixes or the maximal number of populations (generations) is achieved. There is a lot of craftsmanship in definition and assessment of fitness function and the GA parameters. In the presented SBST method there is combination of various constraints and code coverage for estimating the parameters of the evaluation fitness function: statement coverage, branch coverage, condition coverage and toggle coverage. The evolutionary scheme in used GA is very comprehensive and is based on changing advanced strategy $\mathrm{ES}(\mu, k, \lambda, p)$.

The achieved results of testing the DP32 processor core with using 10 different test mixes of length 30 is shown in Fig. 5. In this arrangement, the adaptive feedback loop and evolution method based on GA is used. The constraints, test options and basic parameters for initial setting the test mix generation process are reported in Table 2.

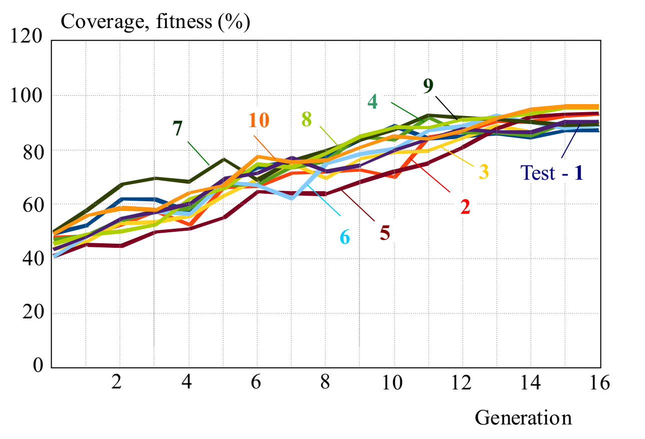

Fig. 5. Code coverage in particular generations of 10 test mixes

Table 2. Initial parameters for the automatic test generation experiment

Parameters	Parameter value
Number of instructions in test mix	30
Number of individuals in generation	10
Number of generations	20
Method for initial population	grouping
Selection method (initial)	tournament
Method GA	GA with feedback
Parameter p (initial mutation)	0,1
Parameter k	∞
Elitizmus	1
Code coverage metric	statement coverage

The automatic generating process produced steady state in code coverage value/fitness by about from 16th generation of test mixes and the final achieved coverage was 95.67 %. The achieved code coverage is sufficient for functional test for processor in comparison with published results till now. Other experiments over this processor shown effectiveness of the proposed SBST method and using new ES with variation based on parameter k and p is very useful in test mixes generation.

5 CONCLUSION

Functional test generation is a long-standing open problem, which is an important problem to be solved for design verification, manufacturing testing and periodic testing processors during their lifetime as well. One key to develop a practical functional test generation approach is to avoid the exponential growth of the test generation complexity in terms of the growth with respect to the design size. The SBST approaches and achieved results over them till now have showed that such functional tests for various processors reached significant level of maturity. They are used not only in academic designs but also in practical SoC and embedded system applications. Many of these SBST research efforts are expected to be intensified on the present and in near future. Application of GAs and evolutionary strategies are representative basis for SBST methods.

The paper has presented contribution to use different operators applied in GAs and varying evolutionary strategies involving also the latest in SBST methods with feedbacks. The main achieved result is the new adaptive and flexible test generation method for verification and testing processors and its automatic implementation. Test programs for processors based on ISA can be generated automatically in shorter time in comparison with the SBST methods where only one ES and random first population are used. Functionality and effectiveness of the new implemented SBST method have been tested on the DP32 processor.

Next research works will be concentrated on other experiments and possible modifications of the presented test generation SBST method, if necessary.

Acknowledgements

This work has been supported by Slovak National Research Grant Agency under project no. VEGA 1/1008/12 Optimization of low-power design of digital and mixed integrated systems, project no. VEGA 1/0616/14 Methods for the design and verification of digital systems with low power consumption using formal specification languages and project no. VEGA 1/0649/09 Security and reliability in distributed computer systems and mobile computer networks.

References

[1] BERNARDI, P.—GROSSO, M.—SANCHEZ, E.—REORDA, M. S.: Software-Based Self-Test of Embedded Microprocessors, In: Design and Test Technology for Dependable Systems-on-Chip, Chapter 15 (R. Ubar, J. Raik, H. T. Vierhaus, eds.), Information Science Reference, IGI Global, Herschey, New York, 2011, pp. 339-359.

[2] PSARAKIS, M.—GIZOPOULOS, D.—SANCHEZ, E.—REORDA, M. S.: Microprocessor Software-Based Self-Testing, IEEE Design & Test of Computers **27,** No. 3 (May/June 2010), 4–19.

[3] KABIRI, P. S.—NAVABI, Z.: Effective RT-Level Software-Based Self-Testing of Embedded Processor Cores, In: 2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, DDECS 2012, Tallinn, Estonia, April 18–20, 2012, pp. 209–212.

[4] BERNARDI, P.—CIGANDA, L. M.—SANCHEZ, E.—SONZA REORDA, M.: MIHST: a Hardware technique for Embedded Microprocessor Functional On/line Self-Test, IEEE Transactions on Computers **PP** No. 99 (2013), 1–12.

[5] SCHÖLZEL, M.—KOAL, T.—RÖDER, S.—VIERHAUS, H. T.: Towards an Automatic Generation of Diagnosis in Field SBST for Processor Components, In: 14th IEEE Latin-American Test Workshop, Cordoba, Argentina, April 3-5, 2013, pp. 1–6.

[6] CORNO, F.—SANCHEZ, E.—SONZA REORDA, M.—SQUILLERO, G.: Automatic Test Program Generation: A Case Study, IEEE Design & Test of Computers **21** No. 2 (Mar/Apr 2004), 102–109.

[7] BERNARDI, P.—SANCHEZ, E.—SCHILLACI, M.—SQUILLERO, G.—SONZA REORDA, M.: An Evolutionary Methodology to Enhance Processor Software-Based Diagnosis, In: Proceedings of 2006 IEEE Congress on Evolutionary Computation, Vancouver BC, July 16-21, 2006, pp. 859–864.

[8] SANCHEZ, E.—SONZA REORDA, M.—SQUILLERO, G. : Efficient Techniques for Automatic Verification-Oriented Test Set Optimization, Int. Journal of Parallel Programming **34** No. 1 (Mar 2006), 93–109, Springer.

[9] SANCHEZ, E.—SCHILLACI, M.—SQUILLERO, G.—SONZA REORDA, M. : An Enhanced Technique for the Automatic Generation of Effective Diagnosis-Oriented Test Programs for Processor, In: Proc. of Design, Automation & Test in Europe Conference & Exhibition 2007, DATE 07, Nice Acropolis, France, April 16-20, 2007, pp. 1–6.

[10] HUDEC, J. : VLSI System Test Design: The Methods, Problems and Experience in Microprocessor Testing Using AFTG, In: Proc. of 16th Int. Conference Information Technology Interfaces ITI94 (V.Ceric, V.H.Dobric, eds.), Zagreb University Computing Centre, Pula, June 14-17, 1994, pp. 191-193.

[11] ModelSim SE Command Reference Manual. Mentor Graphics Corp. 2009. http://www.supportnet.mentor.com/ support/documentation/se/pdf_6.5/modelsim_se_ref.pdf, 2009.

[12] BELKIN, V. V.—SHARSHUNOV, S. G. : ISA Based Functional Test Generation with Application to Self-Test of RISC Processors, In: Proceedings of the 9th IEEE Workshop on Design & Diagnostics of Electronic Circuits & Systems (DDECS 2006), Prague, Czech Republic, Apr 18-21, 2006, pp. 73–74.

[13] THATTE, S. M.—ABRAHAM, J. A. : A Methodology for Functional Level Testing of Microprocessors, In: Proc. 8th International Symposium on Fault-tolerant Computing, Toulouse, 1978, pp. 90–95.

[14] GIZOPOULOS, D.—PSARAKIS, M.—HATZIMIHAIL, M.—MANIATAKOS, M.—PASCHALIS, A.—RAGHUNATHAN, A.—RAVI, S. : Systematic Software-Based Self-Test for Pipelined Processors, IEEE Transactions on Very Large Scale Integration (VLSI) Systems **16** No. 11 (2008), 1441–1453.

[15] APOSTOLAKIS, A.—PSARAKIS, M.—GIZOPOULOS, D.—PASCHALIS, A.—PARULKAR, I. : Exploiting Thread-Level Parallelism in Functional Self-Testing of CMT Processors, In: Proc. of 14th IEEE European Test Symposium ETS 2009, Seville, Spain, 2009, pp. 33–38.

[16] SQUILLERO, G. : MicroGP – An Evolutionary Assembly Program Generator, Genetic Programming and Evolvable Machines **6** No. 3 (2005), 247–263, Springer, New York.

[17] ASHENDEN, P. J. : The VHDL Cookbook, Department of Computer Science, University of Adelaide, Australia, 1990.

[18] SCHWEFEL, H. P.—BÄCK, T. : Evolution Strategies, In Genetic Algorithms in Engineering and Computer Science (Périaux J. and Winter, G., eds.), John Wiley & Sons Ltd, Chichester, 1995.

[19] HUDEC, J. : Some Results in Automatic Functional Test Design for Processors, In: Emerging Trends in Computing, Informatics, Systems Sciences, and Engineering (Sobh, T, Elleithy, K., eds.), Lecture Notes in Electrical Engineering, vol. 151, Springer Science+Business Media B.V., New York, 2013, pp. 965–972.

[20] HUDEC, J. : An Efficient Technique for Processor Automatic Functional Test Generation based on Evolutionary Strategies, In: Proceedings of the 33rd International Conference on Information Technology Interfaces ITI (V. Luzar-Stiffler, ed.), UCC Zagreb & IEEE CP, Cavtat/Dubrovnik, Croatia, June, 27-30, 2011, pp. 527–532.

[21] HUDEC, J. : Processor Functional Test Generation — Some Results with using of Genetic Algorithms, In: Proceedings of the 2nd Eastern European Regional Conference on the Engineering of Computer Based Systems (ECBS-EERC 2011), Bratislava, Slovakia, Sep, 5–6, 2011 (V. Vranić, ed.), IEEE Computer Press, Los Alamitos: IEEE Computer Society, pp. 159–160.

Received 16 September 2014

Ján Hudec received master study diploma and PhD degree at the Slovak University of Technology in Bratislava. He is assistant professor and lecturer at the Faculty of Informatics and Information Technologies Slovak University of Technology in Bratislava and is interested in communications technologies, digital system design, and architecture of computer systems. His research activities cover the digital system design, testing and reliability of logical circuits and processors. He has been author and co-author of more than thirty publications in journals and international conferences.

Elena Gramatová is an associate professor at the Faculty of Informatics and Information Technologies Slovak University of Technology in Bratislava, responsible for courses and research focused to design for testability and dependability of digital systems. She obtained PhD degree in Technical Cybernetics in 1984. She has leaded national projects and participated in seven European framework projects. She has been PC member of conferences as ETS, DDECS, BEC, DSD, FPL, ATS and co-author of 80 peer-reviewed scientific journal and conference papers with more than 70 citations. She is member of the golden core IEEE Computer Society and ETTTC. In year 2007 she received Meritorious awards from IEEE Computer Society based on DDECS activities.