Evolutionary distributed test generation methods for digital circuits

Yu.A.Skobtsov[1], D.E.Ivanov[2], V.Yu.Skobtsov[3]
[1]Donetsk National Technical University, Donetsk, Ukraine;
[2, 3]Institute of Applied Mathematics and Mechanics of NAS of Ukraine, Donetsk, Ukraine.
email: skobtsov@kita.dgtu.donetsk.ua[1] skobtsov@iamm.ac.donetsk.ua[3]
ivanov@iamm.ac.donetsk.ua[2]

Abstract

The genetic algorithms of test generation and fault simulation for digital circuits are considered. The main modes of genetic algorithm parallelization for test generation and simulation are represented - “worker-farmer”, “island model”. The program implementation and computer experiments of proposed methods are discussed.

1 Introduction

Evolutionary computations is the new direction in theory and practice of artificial intelligence, which is strongly investigated in present time. This term is used to generic description of the search, optimizing or learning algorithms based on some formal principles of natural evolutional selection. Among this approaches it can be picked out the main paradigms:

- Genetic algorithms (GA);
- Evolutionary strategy;
- Evolutionary programming;
- Genetic programming.

The differences of these approaches basically consist in the way of goal solution representation and in different set of evolutional operators used in evolutional simulation. All paradigms are widely used for test generation of digital systems.

Classical "simple" GA [1] uses binary strings (for example, 0011101) to represent an individual. Therefore it looks very attractive to use GA techniques for a decision of ATPG problems for DS at structural or functional description levels. So genetic algorithms are successfully used for the test generation of digital circuits [1,2] (from 80th) along with the deterministic structural methods. Several GA-based ATPG systems have been developed such as CRIS, GATEST[1], GATTO [2], DIGATE [1] and others. Experience shows that genetic algorithms give the best results for the circuits, oriented to the data processing, while the deterministic methods more successfully work for sequential circuits with difficult control logic. The parallel version are developed for some systems, for example, GATTO*, GATTO$^+$, CGATTO [3] to increase the fault coverage and decrease the CPU time. Different system use the different parallel computer architectures and various program tools. In this paper we represent parallel GA, which allow extending an effective application GA for this problem. In contrast of known ATPG we use the computing cluster on the base of local intranet.

2 Genetic algorithm of test generation

The purpose of automatic test generation is construction of input sequences of binary sets, which check up any physical defect possible in the process of production (or) exploitation of logical circuits. However there is the enormous number of possible potential physical defects. Some class of faults, which simulate the real physical defects, is usually examined therefore. Thus, fault is the model of (one or a few) physical defects. In practice more frequent all is examined by single stack-at constant faults, test generation for which usually gives satisfactory results (fault coverage) for real faults in the circuits. In the case of necessity (low fault coverage), test can be generated for other types of faults, such as shorts, a transistor is constantly opened (short), delays of propagation of signals, etc. It is known, that the test generation is a NP-difficult task, which means that in worst case it is solved by enumeration of all possibilities [3]. Therefore, in spite of the fact that the great number of serial structural methods of test generation are developed, which for many circuits give good results (high fault coverage), the algorithm parallelization for the tests generation were made an attempt [1].

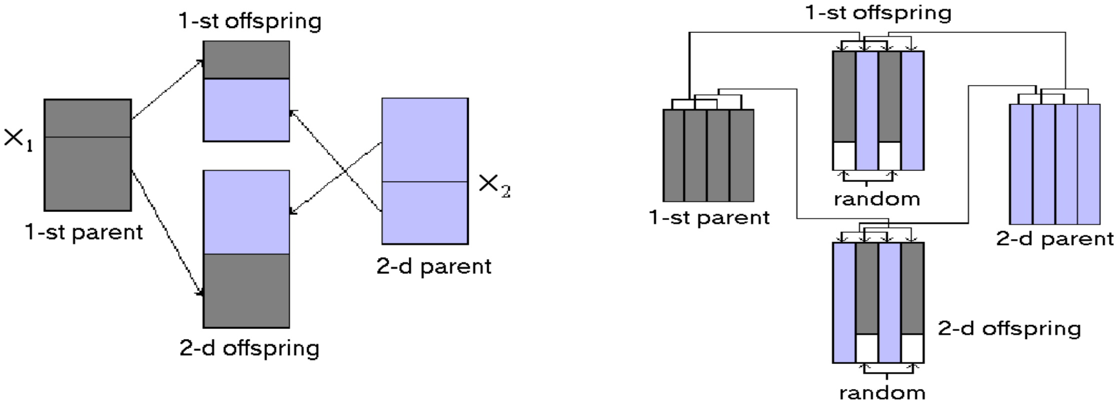
During the test generation for digital circuits with application of GA, as an individual can be used a test sequence represented as binary tables. Population consists of the fixed number of test sequences, possibly, different length.

For the chosen encoding of individuals and populations the following problem oriented genetic operators can be used [2]:

1) **Crossover**. Two types of operation will be realized (Fig.1): vertical and horizontal crossover which are executed accordingly with probabilities P_v and $P_h = 1 - P_v$.

2) **Mutation.** Three types of this operator accordingly are used with probabilities P_{m_1}, P_{m_2} and P_{m_3}:

- Delete of one input vector from the by random chosen position. Application of this operation allows to reduce the length of the generated test sequence in that case, when a remote vector does not worsen test properties of sequence;
- Addition of one input vector in random position, that also allows to extend the search area of decisions;
- Random replacement of bits in a test sequence.



a) horizontal crossover b) vertical crossover

Figure 1: Operations of the horizontal and vertical crossing GA

Because the purpose of test generation is design of input sequence, in which maximally differ the values of signals in fault and good circuits, the quality of test sequence (fitness-function) can be estimated as measure of difference of values of signals in fault and good circuits. In the simplest case the programs of logical simulation of good circuits are used for this purpose, allowing estimating the values of signals on two neighboring (in time) test patterns. On the basis dates of good simulation the following fitness-functions are developed:

$$h(v, f) = c_1 f_1(v, f) + c_2 * f_2(v, f) \tag{1}$$

where - f_1 and f_2 is number of changes of signals on the outputs of logical gates and triggers accordingly ($c_2 >> c_1$).

Fitness function for the sequences determined as the weighed sum of fitness-functions of separate input patterns:

$$H(s,f)=\sum_{i=1}^{length} L^i * h(v_i, f) \qquad (2)$$

where s is the analyzed sequence; v_i - vector from the examined sequence, i - position of vector in a sequence, f - given fault, $0 < L < 1$.

If test generation with mentioned fitness function is not success, then the fault simulation programs are used. Here fitness functions are based on the count of signal difference in good and fault circuits and have approximately the same kind as well as in the previous case.

3 Parallel test generation method based on the “worker-farmer” model

Today numerous modifications and generalizations of GA are suggested [4]. The parallel GAs (PGA) are roughly upcoming and very promising from the theoretical investigation and practical application points of view. Inherent GA "internal" parallelism and possibility of the distributed calculations promote to development of parallel GA. First of all necessary to note that the basis of PGA is population structuring – decomposition to few subpopulations (subsets). This decomposition can be fulfilled with different methods, which define different types of PGA.

In this section for parallelization of GA we use a model «worker-farmer» (“client-server”), because it requires the small changes in the existing software implementation of test generation GA and gives quite good results. In this approach every processor has its own copy of population. The calculation expenses of fitness-functions values (witch use a logical simulation) are evenly distributed to all processors. For all processors the same list of faults is used. Therefore for *n* individuals and *P* processors we take the P/n individuals to every processor. The values of fitness-functions are calculated by the worker processors and are sent to one selected processor-farmer, which collects all information and passes it to all processors. Every processor has information about the values of fitness-function for all individuals and can create next population generation on this basis.

So the processor- farmer executes central part (kernel) of test generation algorithm, while the logical simulation (fault-free and fault) of digital circuits are implemented on processors-slaves. The fault simulation is most critical with point of view of calculation expenses. Different methods of the distributed fault simulation are known, which are mainly based on decomposition: 1) circuits on subcircuits; 2) test sequence on subsequences. We will take combined approach of these two methods.

On the first and second stages the simulated input sequences are distributed between working processors. On the first stage every working processor is loaded by the generation (simulation) of one subsequence. For balance the list of undetected faults is broken up on approximately identical subgroups.

At the end of each of three stages the points of synchronization are placed. When a processor-master arrives at these points, it goes to the wait mode, while all working processors will not make off the tasks that guarantee global correctness of algorithm. Thus work is distributed between a processor-muster and workers as follows.

Processor-master:

- Performs all input-output operations with an user and file system: it reads circuit description and fault list, and writes the generated input test sequence;
- Initially runs «slave» processes on available resources;
- Distributes the copies (internal form) of circuits and fault lists to every working processor;
- Organizes the process control of test generation: as soon as input sequence has to be fault simulated, it sends the proper message for activating of working processors; when working processors finish their work, processor-master receives results and accordingly changes global data structures (general fault list, values of fitness-functions for individuals etc). A processor-worker keeps the local copy of circuit (in internal format) and fault list. Every «worker» takes an input sequence from the « farmer» and determines the faults are detected by this sequence by the logical simulation and calculates the values of fitness-function for individuals. It sends the obtained results to the master and wait next task. As the population size is much larger than a number of processors, good balance in the load of processors is achieved. For every working

processor the change of local fault list with the detected and undetected faults from other working processors requires a lot of resources and it is critical.

Final results (test input sequences and fault coverage) are near to the results obtained on the single possessor computer system with the use of a similar algorithm. Quality of decision (test fault coverage) is not here lost and is in most cases got better, and time of test generation grows short substantially. The data flow diagram of considered algorithm is cited on fig.2.

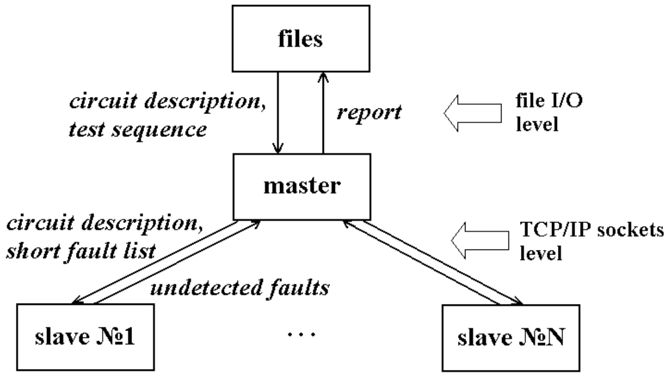


Figure 2: Data flow diagram for distributed test generation and fault simulation algorithms.

4 Distributed fault simulation

Described above distributed genetic algorithm of test generation is based mainly on the distributed fault simulation algorithm. Now we will shortly describe also this method.

One of the central problems of digital device diagnostic is fault simulation of digital circuits. And persistent increasing of modern device complexity makes the task of reducing fault simulation time still actual. One of possible ways to speed up fault simulation procedure is adaptation of existing algorithms for multiprocessors computing systems (clusters) implementation.

Distributed fault simulation is organized in similar way and is based also on the «worker-farmer» approach like distributed test generation. One processor here is selected as farmer (master) and remained processors – as workers (slaves). There exist several approaches to implementation of distributed fault simulation: partitioning of circuit [4] and partitioning of fault list [4]. Our algorithm is based on the fault list partitioning.

The kernel of this process is the procedure *«parallel_fault_simulation»*, which is a regular fault simulator that used in single processor implementation. In our case we used home built PROOFS-based fault simulator, described in [2]. Mark the main advantages of this algorithm that makes it very successful: 1) dynamic fault-list processing: detected fault is eliminated from fault list in the same time it detected, no simulation performs for this fault further; 2) fault sorting which allows include in one group the faults that cause the same simulation events; 3) the technique of functional fault injection.

Master process starts with search procedure of calculation clients. Further it divides full fault list into sublists prorate number of found clients. Then for all clients the following operation sequence is fulfilled in cycle: sends circuit description in internal format; sends test sequence and corresponding short fault list. After that master went in state of waiting of data from clients. At the next step master receives the results of fault simulation from each client and makes general reports: fault coverage, common simulation time, time of simulation on every clients. The constructed in described way distributed fault simulation algorithm allows a high parallelization of simulation process.

5 Distributed test generation based on the "model of islands"

In this section for GA parallelization the "model of islands" is used. Here separate subpopulation, which is initialized randomly and evolved independently, is realized on each processor. In given iteration some subpopulations are exchanged with some individuals in certain way. Each processor select the best individuals of own subpopulation and send them to neighbor processors subpopulations (neighborhood concept is a parameter of method). These individuals are accepted in neighbor processors subpopulations and then independent evolution on each processor-“island” is continued.

In this approach there are more chances to obtained high-quality solution, since different areas of search space are investigated on different processors. Moreover in this case it is possible the reducing of search time due to the best individual migration.

In contrast to previous method (“worker-farmer” model), where GA works only on the central processor-master and processors-slaves are used only for fitness function computing, in this approach full GA is implemented on every processor. In other words each processor executes full cycle of GA evolution operations: fault-free and fault logical simulation, test sequences generation. Each processor works with full circuit and fault list. At the same time there are two reasons of speeding-up test generation process at least. Every processor operates with subpopulation of less dimension that less time is required. Due to the best test sequences migration each processor can detect faults quicker then in case of independent operation in subpopulations. One of the most important parameter of this model is population power (individual number) of subpopulation. The influence and selection of this parameter will be considered below.

6 Implementation and experimental investigations of distributed genetic algorithms of test generation and simulation

Developed algorithms were implemented with using blocking sockets technology in C++ Builder programming environment. For computer experiments the computing cluster on the base of 100 Mbit local intranet was used. The cluster nodes have following parameters: Intel Celeron 2 GHz processor, RAM 256 Mb, OS Windows XP.

For research of effectiveness of suggested algorithms following time parameters were calculated during computer experiments: whole time of simulation process, events number in fault-free and fault simulation, whole number of events. For comparison the experimental results of algorithms from [2] were taken.

At first let consider experimental results obtained for distributed GA implementation based on the “worker-farmer” model. The diagram of simulation speeding up for circuit s35932 (ISCAS89), under condition of change of processors-client number from 1 to 8, is represented on fig.3. Given experimental results confirm the effectiveness of suggested parallelization method of simulation algorithm. Finally on the fig.4 the simulation results for large circuits of ISCAS benchmark are represented. These data show the relative speeding-up with increase of circuit size. It is explained by that fact that for large circuits the expenses of parallelization are reduced compare with fault simulation expenses.

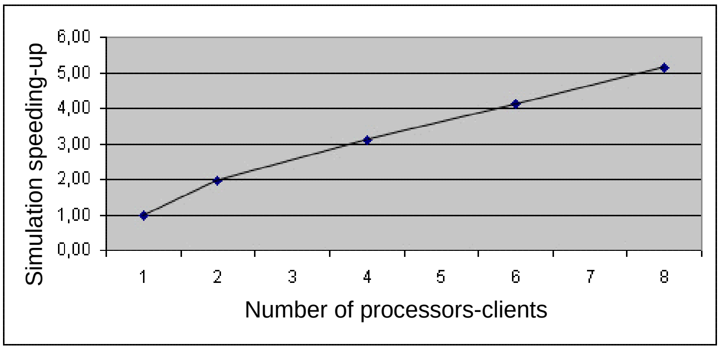


Fig.3. Speeding-up of fault simulation for circuit 35932 according to worker number

Further let consider the results of implementation of test generation with distributed GA, which is based on the “islands model”. In table1 there are represented experimental results, which show the speeding-up and test quality for several circuits from ISCAS89. In this case 8 processors and ring migration method were applied.

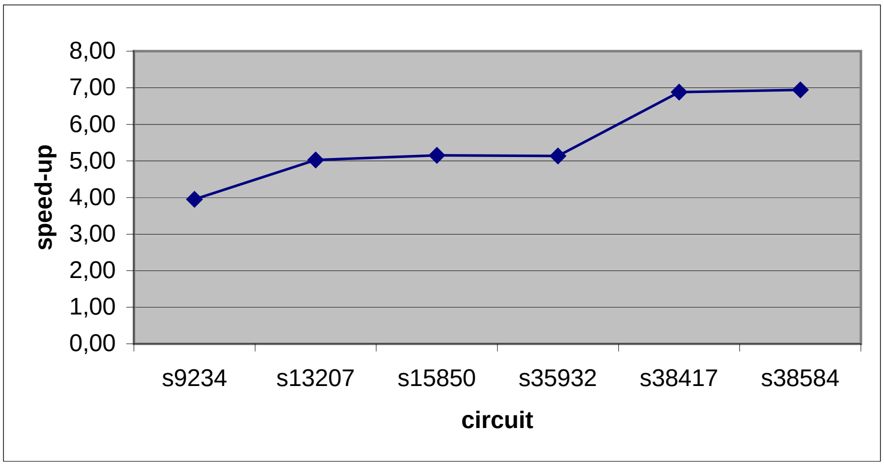


Fig.4. Speeding-up according to circuit complexity

Table 1: Experimental results of implementation of distributed GA-based test generation

Circuit from ISCAS89 benchmark	Speeding-up relatively to one processor	Fault coverage increase
S1196	1.59	+0.8%
S1238	1.8	+0/6%
S1423	1.1	+12.8%
S1488	6.1	+7.1%
S5378	5.16	+1.3%
S35932	5.35	+1.6%

Conclusion

Obtained results confirm the effectiveness of test generation and fault simulation algorithms parallelization. The comparison of experimental results show that “farmer-worker” model gives more speeding-up in comparison with “island model” relatively to one-processor system and essentially easier in software implementation. But “island model” raises fault coverage of generated tests especially for large circuits. Therefore parallelization based on the “island model” has a reason only in case when generated tests have unsatisfactory fault coverage for “farmer-worker” model.

References

[1] P.Muzuder, E.M.Rudnic: Genetic Algorithms for VLSI Design, Layout&Test Automation, Prentice Hall PTR, 1999.

[2] P.Prinetto, M.Rebaudengo, R.M.Sonza: An Automatic Test Pattern Generator for Large Sequential Circuits based on Genetic Algorithms, in: Proceedings of the International Test Conference, 2.-6. October 1994, Washington DC USA, 1994, pp.240-249.

[3] F.Corno, P.Prinetto, M.Rebaudengo, M.Sonza Reorda,E.Veiluva. A PVM Tool for automatic test generation on parallel and distributed systems, in: Springer Verlag: Proceedings of the International Conference on High-Performance Computing and Networking, Milan Italy, 3.-5. May, 1995, pp.39-44.

[4] Y.A.Skobtsov, V.Y.Skobtsov: The Logical Simulation and Testing of Digital Devices, Donetsk:IAMM NASU, 2005. (in Russian).