Distributed Fault Simulation and Genetic Test Generation of Digital Circuits

Skobtsov Y.A.[1], El-Khatib A.I.[1], Ivanov D.E.[2]
[1]Donetsk National Technical University,
[2]Institute of Applied Mathematics and Mechanics NAS of Ukraine
[1]skobtsov@kita.dgtu.donetsk.ua, [2]ivanov@iamm.ac.donetsk.ua

Abstract

Fault simulation is on of the most highly compute-intensive task in the technical diagnostics. One of the ways to speed-up this process is a parallelization on the calculation cluster. In this paper a distributed algorithm for fault simulation of digital circuits is presented. It is based on the well-known «master-slave» approach in which one processor is nominating as a master and rules all calculation on the all slave's processors. To reach the maximal utilization of the processors in the cluster it is used schema with static fault list partitioning.

1. Introduction

One of the central tasks of technical diagnostics – is the problem of high-speed fault simulation of digital circuits. This type of simulation is used for determine the diagnostically properties of test sequences. In this time there are several high-speed algorithms of this fault simulation [1, 2, 3]. All these algorithms for speed-up of process of simulation are used the strategy of dynamical fault list compressing. In this case fault is removed from fault-list in that simulation time in which it was detected and no simulation will perform for this fault on the residuary input patterns.

But the increase of the developed circuit's size keeps the task of fault simulation one of the most actual. One of the possible ways to speed-up of this process is generalization of the existing algorithms to work on the multi processor systems (clusters). This idea can be implemented in two principal different ways. The first one is consist in the assignment one of the processors as a server, which controls the fault list. This processor (named further also as *master*) sends for other processors (named *slaves*) some part of fault list and receives the results of the simulation. The goal of the slave processor is to receive circuit description in some format, also receive fault list, which send by master, and send the simulation results back to the master. This type of algorithms of distributed simulation is proposed in [4, 5], which allows to speed up the fault simulation up to six times on the eight-processors cluster. It is necessary to note that for small circuits the speeding-up is near the one.

It is also was made attempt to construct the distributed algorithms of simulation for clusters with common memory. In these algorithms the simulation is performed by cutting the circuit on the part [6, 7]. But proposed algorithms do not allow the uniform busy of processors in the cluster. Also it is know the papers in which the dependence of interconnection effectiveness is studied of data transmission among the master and slave processors basing on the circuit description [8].

The algorithm that proposed in this paper is based on the fault list partitioning principle and on the delegation for one processor the master functions. As an interchange base it is used TCP/IP protocol. This fact allows include in the calculation cluster the computers, which base on the same or different calculation platforms, for example Windows and Unix. Thus the algorithm of distributed simulation is divided into two independent algorithms. A first algorithm is works on the main processor, named master, and realizes common management function and file input-output. The second algorithm directly executes the fault simulation and works on several processors, which are accessible in the cluster. Each slave processor executes simulation only on some part of full list of faults. Master processor performs the partitioning of full list of faults before the simulation starts.

This paper has the next structure. In the introduction the actuality is substantiate. In the second section we shortly describe an algorithm of parallel fault simulation that developed by authors early. This algorithm is used on the slaves' processors with minimum changes. In the third section we describe the algorithm of distributed fault simulation for multiprocessor calculation system with distributed memory. In the fourth section we describe the calculation experiments and corresponding experimental data and make some remarks about the effectiveness of using of proposed algorithm while working with huge digital circuits. In the next part op paper we describe the way in which distributed genetic test generation algorithm may be built. In the next section we make the conclusions and outline the

further development.

2. Parallel Fault Simulation.

On the each element of the cluster, that is used for simulation as a slave, is realized the parallel fault simulation algorithm with dynamical fault list compression [3]. It shows good experimental characteristic (time of simulation) while working with ISCAS-89 benchmark set [9]. This algorithm was slightly changed in parts which works with input of circuit description, fault list and test sequence. In new version this input is realized not via file I/O but by socket technology, allowing loading the necessary data via local or global networks built on TCP/IP protocol. Also it is necessary to notice that this allows construct the calculation cluster for given algorithm on the computers that works on the operating systems of different type.

Now we shortly describe the algorithm of slave processor. It is destination for fault simulation of single constant (const0 and const1) faults for combinational and sequential synchronous digital circuits. For simulation it is used 3-valued alphabet $E_3=\{0,1,u\}$. Parallel by faults simulation is performed in corresponding with algorithm, which pseudo-code is given on Fig.1.

```
fault_simulation(circuit,test,fault_list)
{
 while ( still_have_input_patterns )
 {
  change_input_pattern();
  simulation_of_fault-free_circuit();
  while(got_faults_for_input_pattern)
  {
   make_fault_group();
   restote_values_for_flip-flops()
   fault_injection_on_primary
   _inputs();
   simulation_of_faulty_circuit_on
   _single_pattern();
   fault_injection_on_primary
   _outputs();
   check_faults_testability();
   store_values_of_flip-flops();
  }
 }
 save_undetected_faults();
}
```

Figure 1. Parallel fault simulation algorithm.

Mark the several key features of this algorithm that essential influence on speed of fault simulation:

- dynamical fault list compression: in this algorithm the fault is removed from fault list at the same modeling time it was detect; in further no simulation for this fault is performed;
- static fault ordering: in this algorithm an preliminary sorting of faults "from primary output in depth" is used, that allows include in the same group the faults that cause the same events in fault simulation;
- functional fault injection: this allows to avoid of unnecessary checks for fault injection for all ports of gates; for gates with fault(s) in this modeling time the sham fault gate is constructed that permits executes the fault injection check only for this type of gates;
- fault flip-flop state saving: for each input pattern we store flip-flop state only in that case if it's different from the same flip-flop state for fault-free circuit.

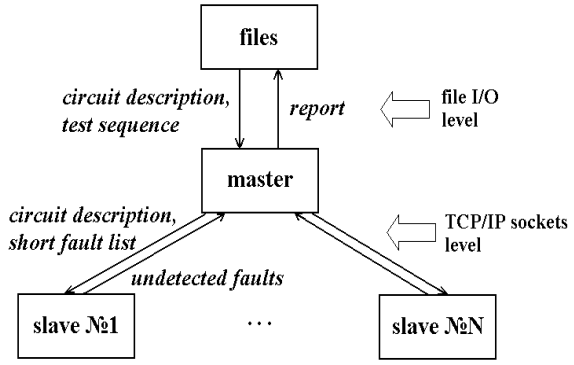


Figure 2. Data flow diagram for distributed fault simulation

3. Distributed Fault Simulation

The algorithm of distributed parallel fault simulation is belongs to the class of algorithms, which for speed-up the simulation process partitioning full fault list into several small lists. The master makes only supervising functions: loads the circuit description, sends this description and corresponding short list of faults for each slave processors, receives the results of simulation from slaves and makes common report.

Each of available clients executes parallel fault simulation on the received data and returns list of undetected faults to master. Now we describe in detail work both master and slave algorithms.

Data flow diagram that describes interconnection among master processor and slaves one is shown on Fig.2. Data transmission in this algorithm is organized in two levels. Master processor for bringing circuits description, full fault list and input sequence uses file input/output. In contrast all data exchange among master and slaves processors is performed with TCP/IP sockets technology. In realization of the proposed algorithm as an interconnection environment was used 100Mbit local network.

The pseudo-code of master algorithm is given on Fig.3. Server starts work from loading circuit description, full list of faults and input test sequence. These procedures are using file I/O. After this socket-master starts and it search via socket for available slaves. If this done successfully then full fault list is partitioned proportionally by number of slave processors. Further for each found slave performs next operations. The description of circuit, part of full list of faults and full test sequence are sent to the slave(s). After this master process is turn into waiting state. Next, master receives list of undetected faults and makes complete simulation report: quality of test sequence, simulation time on each slave processors, time for data exchange with every slaves and full time of simulation.

```
slave_process_fault_simulation()
{
  search_of_master_process();
  if( master_was_found )
  {
    receive_circuit_description();
    receive_short_fault_list();
    parallel_fault_simulation()
    send_list_of_undetected_faults();
  }
}
```

Figure 3. Slave process algorithm

The pseudocode of slave process is given at fig.4. After process start the search of master process takes place. If server is found then client go to data waiting phase. From server client receives the following data: the circuit description, the brief fault list and input sequence. After receiving the necessary data the fault simulation is fulfilled for necessary fault subset. This process is implemented in function "parallel_fault_simulation()". Note that this process is executed accordingly algorithm that is represented in section II. After fault simulation finishing the slave transmits to master the following data: list of undetected faults, work time and information exchange time. After done this client go the waiting phase and ready to receiving the new data for fault simulation.

```
distributed_simulation(circuit,test)
{
  number_of_slaves = search_of_slaves();
  if( number_of_slaves != 0 )
  {
    input_circuit_description();
    input_test();
    make_full_fault_list();
    partitioning_fault_list(number_
    of_slaves);
    for(i=0;i<number_of_slaves;i++)
    {
      send_to_client_i_circuit
      _description ();
      send_to_client_i_part_
      of_fault_list();
      send_to_client_i_test_sequence();
    }
    for(i=0;i<number_of_slaves;i++)
    {
      receive_list_of_undetected
      _faults();
    }
    make_report();
  }
}
```

Figure 4. Master process algorithm for distributed simulation

4. Realization and Experimental Data

The proposed algorithm of distributed fault simulation is implemented in C++ Builder software environment using blocking socket technology. This technology assumes that calculation process in points of information exchange must be stopped until the transmitting side sends the necessary data.

The server process program listing is about of 1300 statements C++ and based on algorithm from [3] with insignificant modification. In particular the file input-output of circuit description is changed to network exchange with using of blocking socket technology. The report transfer to computer cluster is implemented with using the same technology.

For test operation it used the computer cluster based on local network 100Mbit/sec of the usual educational class. The cluster components have following characteristics: Intel Celeron Processor 2000Mhz, 256MB RAM, operational system Windows XP.

For the purpose of research for the effective application of the proposed algorithm during computer experiment the following time characteristics were evaluated: the total time of the simulation process, the number of events during the fault and fault-free, and the total events number.

For comparison we choose the characteristics of algorithm described in [3] on the personal computer with corresponding configuration.

The experimental data for medium-size circuit S9234 are given in table 1. This circuit has the following characteristics: input number –19, output number –22, D-trigger number – 228, invertors number - 3570, the other types gate number –2027, the total element number – 5866.

In table 1 the computer speeding-up of multiprocessor realization is shown in brackets in comparison with single-processor realization. The analysis shows that event number during fault simulation practically does not increase (factor ≈1). It

says about zero-redundant fault simulation in proposed algorithm. On the other hand the event number during good simulation increases with factor near processors number from 1.0 to 7.92 with processor-client rise from 1 to 8. It is due to need of good simulation at all processor-clients. The total events number was increased unessential: factor practically does not change from 1.00 to 1.02 with increasing processors number from 1 to 8. It is explained by fact that event number during good simulation is less 1% of total events number.

At Fig.5 the speed-up of simulation is represented with processor number increasing from 1 to 8 for S9234 circuit.

The shown experimental data validate the effectiveness of proposed method for simulation parallelization. However the completely linear rise of speeding-up is unachievable. The basic obstacle is need of multiplex sending of circuit description and redundant good simulation at each client processor.

5. Distributed GA.

Inherent GA "internal" parallelism and possibility of the distributed calculations promote to development of parallel GA (PGA). The first papers in this direction appeared in 60-th years, but only in 80-th years, when accessible facilities of parallel realization were developed, the PGA researches adopted systematic mass character and practical orientation. The great number of models and realizations are developed in this direction, some of which are represented below [2].

Parallelism of GA gives the following advantages:

1) Search of alternative decisions of the same problem;
2) Parallel search from different points in decision space;
3) Good realization is assumed as islands or cellular structure;
4) Large efficiency of search even in the case of realization not on parallel calculation structures;
5) Good compatibility with other evolutional and classic procedures of search;
6) Substantial increase of speed execution on the multi-possessor systems.

Further we shall consider the modern main methods of the PGA realization. Most known is global parallelism, which represented on Fig.1.a).

This model is based on simple (classic) GA in which the calculations are performed in parallel.

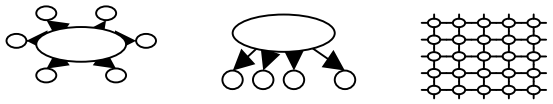
This approach is faster, than classic GA, which can be executed sequentially, and does not usually require balance on the load as on different processors more frequent than all the values of fitness-functions for different individuals (strings) are calculated (having about equal computation complexity). The exception makes the genetic programming, where different individuals can strongly differ on the complication (treelike or graph- structures).

This model often named "**master-slave**". Many researchers use the pool of processors for the increasing of speed of algorithm execution. At the same time the independent program passages of algorithm at different processors are executed essentially quick than at one processor. It must be noted, that in this case there is no co-operation between different passes of algorithm. It is extraordinarily simple method of implementation of simultaneous work (if it is possible) and it can be very useful. For example, it can be used for the solving of the same task with different initial conditions. By virtue of the probabilistic nature GA allows effectively using this method. At the same time we have minimum program changes, but advantages can be considerable.

In Fig.7.b) also represented an extraordinarily popular "**model of islands**" (coarse grain), where great number of sub algorithms simultaneously work in

Table 1. Experimental data for circuit S9234.ben

Description	single-processor realization	multy-processor realization (number of processors)					
		1	2	3	4	6	8
test length	1000	1000					
fault simulation time, sec.	330	336 (0,98)	194 (1,7)	138 (2,39)	107 (3,08)	86 (3,83)	79 (4,17)
number of events, mill.	441,51	440,05 (1,00)	441,81 (1,00)	443,21 (1,00)	443,78 (1.01)	447,79 (1,01)	449,93 (1,02)
number of events of fault-free simulation, mill.	0,48	0,48 (1,00)	0,95 (1,98)	1,42 (2,96)	1,90 (3,96)	2,85 (5,93)	3,80 (7,92)
number of events of fault simulation, mill.	441,03	439,58 (1,00)	440,86 (1,00)	441,79 (1,00)	441,88 (1,00)	444,94 (1,01)	446,13 (1,01)



a) b) c)

Figure 7. Different realization of parallel GA

parallel, exchanging in the search process by some individuals. This model assumes direct realization on the computing systems with MIMD architecture. Thus every "island" corresponds to its processor.

In cellular GA (fine grain), shown on Fig.7.c, parallelism usually will be realized on the computer systems with SIMD architecture, where every processor represents subpopulation (from one individual). Although another papers are known, where authors use single possessor computers and systems with MIMD-architecture.

6. Parallel Genetic Algorithm of Test Generation

In this paper for pararallelism of GA we use a model «master - slave», because it requires the small changes in the existent version of software realizing GA of test generations and gives quite good results.

In this approach every processor has its own copy of population. The cost of calculation of values of fitness-functions (witch use the logical simulation) is evenly distributed on all processors. For all processors the same list of faults is used. Therefore for n individuals and P processors we take to every processor the P/n individuals. The values of fitness-functions are calculated by the slave processor and are sent to one selected processor (master), 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 design next generation on this basis.

So the processor-master executes central part (kernel) of test generation, while the logical simulation (good and fault) of digital circuits will be realized on processors–slaves. With point of view of cost calculation the fault simulation is most critical. Different methods of organization of the distributed fault simulation, which are known, mainly based on breaking-up: 1) circuits on sub circuits; 2) test sequence on a subsequence. We will take combined approach combining these two methods.

On the first and second stages the generated input sequences are distributed between working processors. On the first stage every working processor is loaded by the generation 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, he passes to the wait mode, while all working processors will not make off the tasks, that guarantees global correctness of algorithm. Thus work between a processor-master and workers is distributed as follows.

Processor-master:

- Performs all input-output operations with an user and file system: it reads circuit description and fault list, then, it writes the generated input test sequence;
- Initially spans «slave» processes on available procedures;
- Distributes the copies (in internal form) of circuits and fault lists to every working processor;
- Organizes the control process of test generation: as soon as input sequence has to be fault simulated, it sends the proper message fop 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 and, etc.).

A working processor keeps the local copy of circuit (in internal format) and fault list. Every «worker» takes an input sequence from the «master» and by the logical simulation determines the faults, which are detected by this sequence; also it calculates the values of fitness-function for individuals. It sends the got results to the master and wait next job. Because the population size is much larger than the 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 enough a lot of resources and it is critical.

Final results (test input sequences and fault coverage) are near to those, which are got on the single possessor computer system with the use of a similar algorithm. Quality of problem solution (fault coverage of test sequence) is not here lost and in most cases got better, and time of test generation grows short substantially.

CONCLUSIONS

In this paper a problem of distributed genetic test generation and fault simulation is studied. The possible way of organization of this process is described. Proposed by authors for solving these tasks algorithms that based on the scheme «master-slave» are described.

7. References

1. Niermann T.M., Cheng W.-T., Patel J.H. PROOFS: A Fast, Memory-Efficient Sequential Circuits Fault Simulator // IEEE Trans. CAD. – 1992.– P.198-207.
2. Kung C.P., Lin C.S. HyHope: A Fast Fault Simulator with Efficient Simulation of Hypertrophic Faults // Proc. of International Test Conference. - 1994. - P.714-718.
3. Ivanov D.E., Skobtsov Yu.A., Parallel Fault Simulation for sequential circuits // Artificial Intelligence. – 1999. - №1. – C.44-50.
4. P.A. Duba, R.K. Roy, J.A. Abraham and W.A. Rogers, "Fault simulation in a distributed environment", in Proceedings of the 25th ACM/IEEE Design Automation Conference, pp.686-691, June 1988.
5. T. Marcas, M. Royals and N. Kanopoulos, "On distributed fault simulation", IEEE Computer, vol. 7, pp. 40-52, Jan. 1990.
6. S. Patil, P. Banerjee and J. Patel, "Parallel test generation for sequential circuits on general purpose multiprocessors", in Proceedings of the 28th ACM/IEEE Design Automation Conference, (San Francisco, CA), June 1991.
7. S. Ghost, "NODIFS: a novel, distributed circuit partitioning based algorithm for fault simulation of combinational an sequential digital design on loosely coupled parallel processors", tech. rep., LEMS, Division of Engineering, Brown University, Providence, RI, 1991.
8. Ladyzhensky Yu.V., Popov Yu.V. A Program system for synchronization protocol investigation under distributed logical simulation // Proc. of Donetsk State Technical. University, Series "Computers and Automation".- 2004.- Vol №74.- C.201-209.
9. Brgles F., Bryan D., Kozminski K. Combinational profiles of sequential benchmark circuits // International symposium of circuits and systems, ISCAS-89. – 1989. – P.1929-1934.

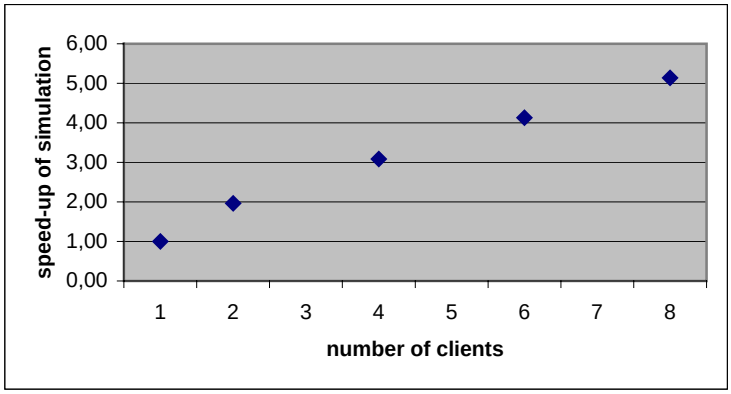


Figure 5. Speed-up of fault simulation for s35938 benchmark circuit depending on the number of the client processors

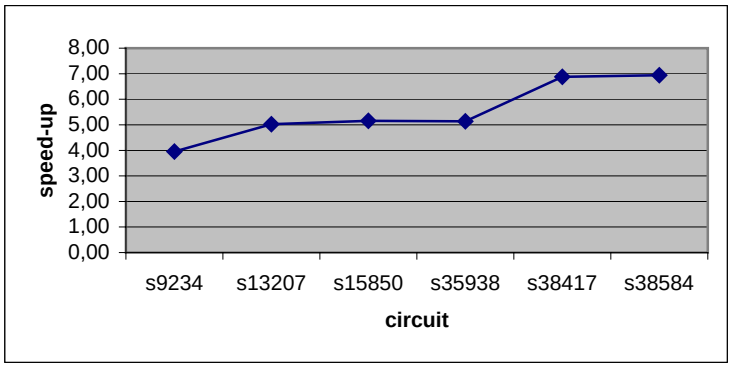


Figure 6. Speed-up of fault simulation for larges ISCAS-89 circuits with 8 clients realization.