UDC 681.518

D.E. IVANOV

Institute of Applied Mathematics and Mechanics NAS of Ukraine, Donetsk

PARALLEL FAULT SIMULATION ON MULTI-CORE PROCESSORS

In this paper we propose a fault simulation algorithm that utilizes all cores in multi-core processors. We adapt for multi-core workstation our early proposed distributed fault simulation algorithm. Proposed algorithm uses multi thread execution. The algorithm is based on the well-known «master-slave» approach in which one thread is nominated as a master and controls the calculation on all the other cores of processor. To maximize utilization of the cores a scheme with static fault list partitioning is used. The speed-up coefficient of the simulation time obtained during machine experiments is up to 3.44 times on the quad core system.

Key words: *digital circuit, sequential circuit, fault simulation, parallel simulation, multi-core processor, execution thread.*

Introduction

Fault simulation is one of the most computation-intensive tasks in technical diagnostics. Main goal of fault simulation is to determine the quality of the input sequence. With the increase of the circuit's size several approaches were developed to speed up the simulation time.

1. Event-driven simulation [1] that enables to simulate only small part of the circuit in which activity is performed contrary all circuit.

2. Parallel simulation where in each bit of the processor word is simulated own copy of modified circuit [2-4]; this approach was of prime importance for fault simulation that need a huge amount of processor time. All of the abovementioned algorithms used the strategy of parallel fault propagation. Also many optimizing approaches are known that allow further decrease of simulation time (for example dynamic fault compressing).

But the fault simulation problem remains one of the most important. One of the possible ways to speed-up this process is generalization of the existing algorithms to work on multi processor systems (clusters) and multi-core workstations. So we have third approach.

3. Distributed simulation. Here two main approaches are possible:

– circuit fragmentation: each part of circuit is modeled on its own processor in cluster [5]; here it is necessary to construct synchronizing protocols [6];

– fault list fragmentation [7-8]; it is used for fault simulation. An algorithm of this type was reported earlier by the author [9]. Experiments were done in regular 100Mbit local intranet. The speed-up coefficient was reported to be up to 7.92 for big circuits on 8 PC in local network.

In this paper an attempt is made to modify our previously developed algorithm to use on multi-core workstation.

Our investigation has three stages. At first we run the existing algorithm on calculation cluster. At the second stage we run the same algorithm on quad-core workstation. Further we eliminate redundant phase of algorithm that carries out circuit data exchange among the processors in cluster (circuit description, fault list, input sequence) and again run it on quad-core workstation. Then we compare the results obtained on all the stages.

This paper has the following structure. In the first section we underline actuality of problem. In the second section an algorithm of distributed fault simulation for the calculation cluster is described briefly as a background. An adaptation of this algorithm for multi-core processor systems is described in section 3. In section 4 we describe evaluated experiments in the different environments and compare obtained results. In the section 5 we make the conclusions and outline further development.

1. Distributed fault simulation

The goal of our fault simulation algorithm is to measure the quality of a given input sequence, particularly fault coverage for single stuck-at faults. The main idea of algorithm is to divide list of faults to be simulated into several small lists. Complete fault list is partitioned proportionally to the number of slave processors. Then each of these small lists of faults is simulated on own computer. Figure 1 shows the interaction between master and slave computers. Figure 2 shows pseudo-code of master algorithm. On client's

processors a home-built PROOFS-like fault simulation algorithm [4] is implemented.

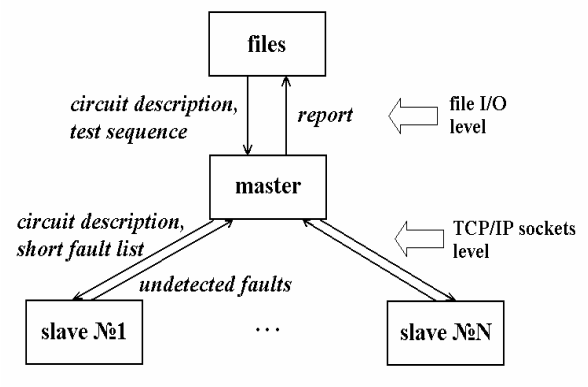

Fig. 1. Data flow diagram for distributed fault simulation

```
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_data();
    }
    for(i=0;i<number_of_slaves;i++)
    {
      receive_list_of_undetected
      _faults();
    }
    make_report();
  }
}
```

Fig. 2. Master process algorithm for distributed simulation

2. Modification for multi-core processors

The algorithm described in previous section is oriented on the cluster calculation environment. In this system each processor has its own local memory and complete data exchange is performed using interconnection network and communication protocol (in current case TCP/IP). Figure 2 underlines this aspect by picking out the procedures of data exchange: "send_to_client_i_data()".

While adapting this algorithm for multi-core system it is necessary to take into account that all cores have common system memory. So the abovementioned procedures can be eliminated and replaced by passing only pointers to local data (circuit description, input sequence, fault list). The procedure of looking for clients can be eliminated as well.

Each client procedure is organized as a regular calculation thread. Varying the number of client threads we can estimate speed-up coefficient. It is expected that the biggest coefficient of speed-up of the simulation time should be achieved when we choose the number of simulation threads equal to the number of processor cores.

3. Experimental results

We carried out three types of experiments.

Case 1. At first we run our algorithm on local network with homogeneous computers. In first experiment we use local network with 100Mbit/s speed and Intel Celeron 2,0Ghz processors and 256Mbyte system memory.

Case 2. Next we start the same algorithm on quad-core PC. We use ordinary PC workstation with Intel Core 2 Quad E6600 2,4Ghz CPU and 2Gbyte of system memory. Notice that this quad-core CPU in fact contains two dual-core processors that layout on the one chip.

Case 3. Finally we test our algorithm on the same quad-core PC with modifications described in previous section (circuit description transfer was eliminated).

All approbation was performed on the ISCAS-89 circuits [10] with more than thousands logical gates. In all cases we use previously random generated test sequences with 1000 input vectors.

To compare speed-up coefficient of our algorithm in mentioned three experimental environments we choose medium-size circuit s9234ben. Figure 3 shows speed-up of the simulation process in all our experimental cases. We don't report the simulation time because workstations in case 1 and cases 2-3 are not homogeneous. It is necessary note that we have essential fall of performance in case 2, which is caused by redundant data exchange on TCP/IP protocol.

This produced unnecessary CPU utilization. Situation is improved in case 3 where this data exchange eliminated. But the performance grows approximately only up to case 1 level. Obviously the simulation threads create the throng in the common cores cache. Figure 4 shows the speed-up of the simulation time depending on the number of thread of simulation in cases 2 and 3. As is expected maximum acceleration was achieved in cases with four simulation threads. It is obvious that the utilization of CPU's cores in this case is maximal. This fact also confirms the system monitor information. So further in case 3 we use for experiments four simulation threads.

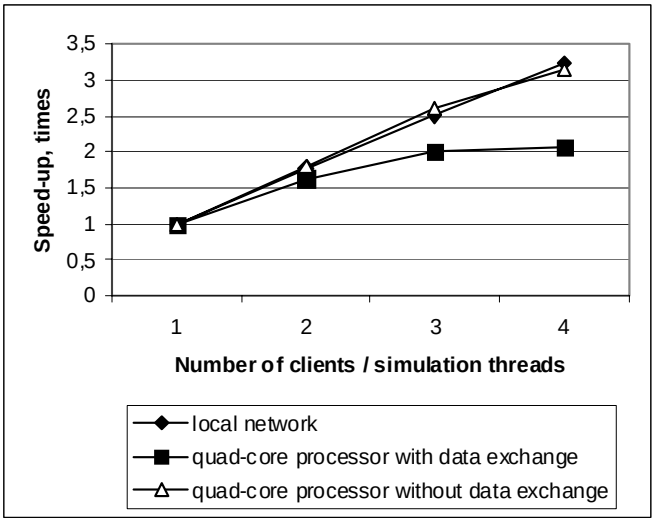

Fig. 3. Speed-up of the simulation time

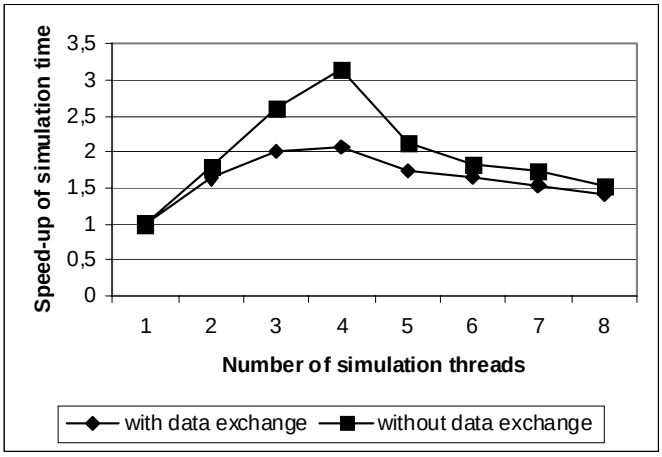

Fig. 4. Speed-up of the simulation time depending on the simulation threads number

Also in Table 1 we bring the speed-up coefficient for different benchmark circuits in environment 3. The speed-up coefficient is varying between 2.81 and 3.44 times.

Table 1

Experimental data for ISCAS-89 circuits

circuit	time of simulation, sec., 1 thread	time of simulation, sec., 4 threads	speed-up, times
S9234	104	33	3.15
S13207	303	88	3.44
S15850	390	135	2.89
S35932	886	315	2.81
S38417	2473	781	3.17

Conclusions

In this paper the problem of utilizing of all cores performance in multi-core PC is discussed. We propose an effective algorithm for fault simulation of digital circuits on multi-core workstation.

This algorithm allows to speed-up the fault simulation time up to 3.44 times for large benchmark circuits. Also in further investigation we expect that the chosen approach allow effective scalability of algorithm with further increasing of the number of processor's cores.

References

1. Breuer M.A. Diagnosis and reliable design of digital systems. M.A. Breuer, A.D. Friedman / Potomac Computer Sc. Press. - 1976. – 308 p.

2. Niermann T.M. PROOFS: A Fast, Memory-Efficient Sequential Circuits Fault Simulator. / T.M. Niermann, W.-T. Cheng, J.H. Patel // IEEE Trans. CAD. – 1992.– P. 198-207.

3. Kung C.P. HyHope: A Fast Fault Simulator with Efficient Simulation of Hypertrophic Faults. / C.P. Kung, C.S. Lin // Proc. of International Test Conference. - 1994. - P. 714-718.

4. Ivanov D.E., Parallel Fault Simulation for sequential circuits. / D.E. Ivanov, Yu.A, Skobtsov // Artificial Intelligence. – 1999. - № 1. – C. 44-50.

5. Ghost S. NODIFS: a novel, distributed circuit partitioning based algorithm for fault simulation of combinational an sequential digital design on loosely coupled parallel processors / S. Ghost // tech. rep., LEMS, Division of Engineering, Brown University, Providence. - RI. - 1991.

6. Ladyzhensky Yu.V. A Program system for synchronization protocol investigation under distributed logical simulation. / Yu.V. Ladyzhensky, Yu.V. Popov // Proc. of Donetsk State Technical. University, Series "Computers and Automation". - 2004. - Vol №74. - C.201-209.

7. Marcas T. On distributed fault simulation. / T. Marcas, M. Royals, N. Kanopoulos // IEEE Compute. - Vol. 7. -1990. - P. 40-52.

8. Duba P.A. Fault simulation in a distributed environment / P.A. Duba, R.K. Roy, J.A. Abraham, W.A. Rogers // Proc. of the 25th ACM/IEEE Design Automation Conference. – 1988. - P. 686-691.

9. Skobtsov Y.A. Distributed Fault Simulation and Genetic Test Generation of Digital Circuits / Yu. A. Skobtsov, El-Khatib, D.E. Ivanov // Proceedings of IEEE East-West Design&Test Workshop(EWDT'06). - 2006:- Sochi.

10. Brgles F. Combinational profiles of sequential benchmark circuits / F. Brgles, D. Bryan, K. Kozminski // International symposium of circuits and systems, ISCAS-89. – 1989. – P. 1929-1934.

Поступила в редакцию 19.01.2009

Рецензент: д-р техн. наук, проф., проф. кафедры автоматизированных систем управления Ю.А. Скобцов, Донецкий национальный технический университет, Донецк, Украина.

ПАРАЛЕЛЬНЕ МОДЕЛЮВАННЯ ЦИФРОВИХ СХЕМ З ПОШКОДЖЕННЯМИ НА БАГАТОЯДЕРНИХ СИСТЕМАХ

Д.Є. Іванов

В статті пропонується алгоритм моделювання цифрових схем з пошкодженнями, який розраховано на використання в обчислювальних системах з багатоядерними процесорами. Даний алгоритм є адаптацією для багатоядерних систем розподіленого алгоритму моделювання цифрових схем з пошкодженнями, який було запропоновано авторами раніше. Даний алгоритм використовує багатопотокове виконання. Алгоритм базується на вже відомому підході "хазяїн-працівник", при якому один виконавчий потік призначається головним та слідкує за розподіленням обчислень по ядрах процесору. Для підвищення коефіцієнтів завантаження ядер процесору ми використовуємо схему з розбиванням списку пошкоджень. Обчислювальні експерименти, що були проведені, показують збільшення коефіцієнту прискорення процесу моделювання до 3,44 разів на системах з чотириядерним процесором.

Ключові слова: цифрова схема, послідовністна схема, моделювання з пошкодженнями, паралельне моделювання, багатоядерний процесор, виконавчий потік.

ПАРАЛЛЕЛЬНОЕ МОДЕЛИРОВАНИЕ ЦИФРОВЫХ СХЕМ С НЕИСПРАВНОСТЯМИ НА МНОГОЯДЕРНЫХ СИСТЕМАХ

Д.Е. Иванов

В статье предлагается алгоритм моделирования цифровых схем с неисправностями, который рассчитан на использование на рабочих станциях с многоядерными процессорами. Данный алгоритм является адаптацией для многоядерных систем ранее предложенного авторами распределённого алгоритма моделирования цифровых схем с неисправностями. Предлагаемый алгоритм использует многопоточное исполнение. Алгоритм основан на ранее описанном подходе «хозяин-рабочий», при котором один исполнительный поток назначается в качестве основного и контролирует распределение вычислений по ядрам процессора. Для повышения загрузки вычислительных ядер процессора мы используем схему моделирования с разбиением списка неисправностей. Проведенные машинные эксперименты показывают коэффициент ускорения времени процесса моделирования до 3,44 раз для систем с четырёхядерным процессором.

Ключевые слова: цифровая схема, последовательностная схема, моделирование с неисправностями, параллельное моделирование, многоядерный процессор, вычислительный поток.

Иванов Дмитрий Евгеньевич – канд. техн. наук, доцент, старший научный сотрудник отдела теории управляющих систем Института прикладной математики и механики НАН Украины, Донецк, e-mail: ivanov@iamm.ac.donetsk.ua.