Distributed fault simulation for sequential circuits by pattern partitioning

W.-C. Wu
C.-L. Lee
J.-E. Chen
W.-Y. Lin

Indexing terms: Distributed machines, Multiprocessors, Pattern partitioning

Abstract: The paper investigates distributed fault simulation by pattern partitioning for sequential circuits. Simulation is done by making each distributed machine perform fault-free simulation with preceding patterns and then perform fault simulation with its own patterns. The fault simulation is accelerated since the number of patterns needed to be performed fault simulation for each machine is reduced by a factor of n, the number of machines, and the faults detected by any machine are dropped through communication of the network. A superlinear speedup can be obtained because this method can automatically remove the Case 1 faults, which are time consuming faults and would be considered to be undetected in the traditional three-valued fault simulation but are in fact truly detected. A mathematical model is also presented to predict the performance of the distributed fault simulation.

1 Introduction

Many fault-simulation algorithms such as parallel [1], deductive [2], concurrent [3], critical path tracing [4], PROOFS [5], PARIS [6], COMBINED [7], HOPE [8, 9] and single-event equivalence [10] fault simulation have been developed to accelerate the speed of fault simulation. However, all the above algorithms have the computation complexity between $O(G^{1.5})$ and $O(G^3)$, where G is the number of gates of the simulated circuit. One solution is the hardware approach. Much work has implemented the fault simulation in special-purpose hardwares such as IBM's Yorktown Simulation Engine (YSE), NEC's Hardware Accelerator (HAL), Zycad, Silicon Solutions, Daisy, AT&T's Microprogrammable Accelerator for Rapid Simulation (MARS) [11–15]. However, these machines are much more expensive than a general purpose machine and their flexibilities are very poor because of their fixed architectures. An alternative type of hardware approach is to make use of general purpose machines such as workstations which are connected through a network and the work of a fault simulation job is distributed to these workstations to execute concurrently.

To partition a fault simulation job for execution on a distributed workstation system there are three types of partitionings: circuit partitioning, fault partitioning, and pattern partitioning. Circuit partitioning for sequential circuits was reported on special cluster multiprocessors [16, 17], i.e. an Encore Multimax shared memory multiprocessor or an INTEL iPSC/2 hypercube with a Sequent Balance 21 000 shared memory machine. The problems of fault partitioning are how to group independent faults which cannot be detected by the same pattern, and how to partition the fault list in a homogeneous workload. Presenting a circuit in a hierarchical way can guide to group tolerant faults by using the hierarchical structural circuit representation [18]. To solve the unbalanced workload, dynamic allocation of the faults has been used in relevant works [19–21], but this scheme takes more communication time because the faults and the states of the circuit must be transmitted. Pattern partitioning was reported only for combinational circuits [22, 23] since these approaches could not solve the problem of order correlations in the test patterns for sequential circuits.

In this paper, we target a single stuck_at fault model and the pattern partitioning on the distributed fault simulation because the number of patterns can be reduced by a factor of n, the number of machines, and the faults detected by any machine can be dropped through communication of the network. The simulation is done by making each distributed machine perform fault-free simulation with preceding patterns and perform fault simulation with its own patterns. A superlinear speedup can be obtained because this method can automatically remove Case 1 faults, which are time consuming faults and would be considered to be undetected in the traditional three-valued fault simulation but are truly detected in fact. In addition, an analytic model is presented to describe the fault-simulation process for both the non-partitioning and multipartitioning processes. The model can be used to predict the speedup and efficiency for pattern-partitioning fault simulation.

Paper 1867E (C3, E10), first received 15th August 1994 and in final revised form 16th January 1995

W.-C. Wu, C.-L. Lee and W.-Y. Lin are with the Department of Electronics Engineering & Institute of Electronics, National Chiao Tung University, 1001 Ta-Hsueh Rd., Hsin-Chu, Taiwan, Republic of China

J.-E. Chen is with the Department of Electrical Engineering, Chung-Hua Polytechnic Institute, 30 Tung-Shiang Rd., Hsin-Chu, Taiwan, Republic of China

This work was supported by the National Science Council, Republic of China, under grant NSC-82-0404-E009-183

2 Architecture and procedure for pattern partitioning for fault simulation

Fig. 1 illustrates the architecture for the distributed fault simulation adopted in this paper. Every ellipse represents

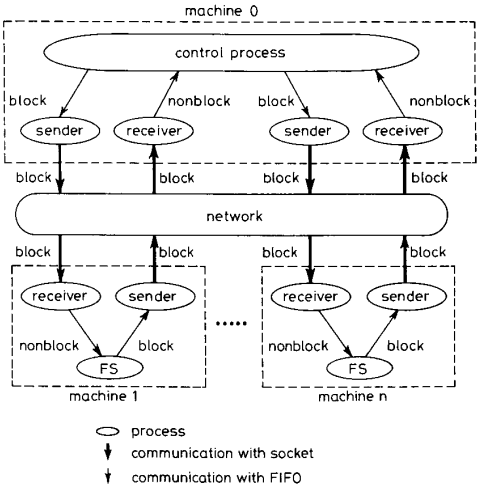


Fig. 1 *Architecture of distributed fault simulation*

a process, which is an instance of a program that is being executed by the operation system. Socket is used for communication between processors in different machines and FIFOs in the same machine in UNIX. The data can be transferred in both the block or nonblock types. For block type of communication, the system will wait until the data on the Socket or FIFO are available to ensure that no data are lost. For the nonblock type communication, the system checks whether the data are available or not and gets the data if they exist, otherwise it executes the next instruction. This communication technique can reduce the communication time because the system only gets the available data and thus there is no waiting time.

The fault simulator used in this work was SEESIM [10], which is a fast sequential circuit fault simulator based on single-event equivalence. For a sequential circuit there is dependence between test patterns. The state of the circuit depends on the previous patterns applied to the circuit. When the patterns are partitioned into several groups for distributed fault simulation, each machine needs to simulate the circuit into the same state of its previous machine to obtain the right results. Hence, in this work, each machine must firstly perform the fault-free simulation with the patterns that are performed fault simulation by the previous machines.

The procedure for each machine is briefly described as follows

```
begin
  {receive the numbers of the fault-free simulation
    patterns and the fault simulation patterns from
    the control process;}
  {do the fault-free simulation with the fault-free
    pattern set;}
  while (the fault simulation pattern set is not empty)
    get the next pattern;
    do the fault simulation;
    [send detected faults to the control process;]
    [drop detected faults received from the control
      process;]
  end while
  send the results to the control process;
end
```

The procedures braced by { } is inserted for receiving the patterns and driving the circuit to the starting state. Next it performs fault simulation until the fault-simulation pattern set is empty. During this while-loop of fault simulation, the procedures braced by [ ] are inserted to drop faults externally by communication with other machines. Communication period λ was defined that each processor transmits the detected faults every λ patterns [24]. In the procedure we reduced the communication time by using the technique of the dynamic communication period. In this dynamic adjustment, λ is small for the first several patterns because they are very effective in detecting faults, and for the latter patterns, which detect fewer faults, λ is large.

In Fig. 1 the control process performs the jobs of interfacing with the user, creating the processes needed, transmitting detected faults, and reporting the results.

3 Speedup estimation

A very simple mathematical fault-simulation time model was formulated to study how fault simulation can be accelerated by using the distributed simulation from a theoretical point of view. Fig. 2*a* and 2*b* show the simulation times against the simulated pattern for s5378 for a one-machine simulation and two-machine simulation without communication, respectively. The fault-simulation time curves for other circuits exhibit similar curves as that in Fig. 2. The simulation curve of Fig. 2*a* can be curve fitted by the following equation

$$T_s(x) = \alpha x^{-r} \tag{1}$$

where x is the pattern number, α is the simulation time of the first pattern, and r is a decay index whose value is $0 < r < 1$. For s5378, $\alpha = 46.89$ (sec/60), and $r = 0.214$. Hence, the total simulation time T_t is

$$T_t = \int_1^P \alpha x^{-r}\, dx \tag{2}$$

where P is the number of patterns. In Fig. 2*b*, the simulation curve of the first machine simulation is the same as that of the one-machine simulation but the second machine simulation curve has a similar shape of the first machine curve while has a different α and r. An expanded depiction of this part of the curve is shown in Fig. 2*c*, where a fitted curve of $T_s = 17.78\, x^{-0.156}$ is also shown.

In the preceding description, if there are communications between two machines during the simulation process to drop faults, the two curves for two machines will be identical and their α and r will be lower than that of Fig. 2*b*. For n-machine distributed fault simulation with communication, the simulation time curve for each machine can be written as

$$T_{s_n}(x) = \alpha_n x^{-r_n} \tag{3}$$

where α_n and r_n are parameters of n-machine simulation. Hence, the total simulation time T_{t_n} will be

$$T_{t_n} = T_0 + \int_1^{P/n} \alpha_n x^{-r_n}\, dx \tag{4}$$

where T_0 is the fault-free simulation time. In our simulation results, the machine that had lastly done its job was almost always machine 1. T_0 for machine 1 is 0. So the

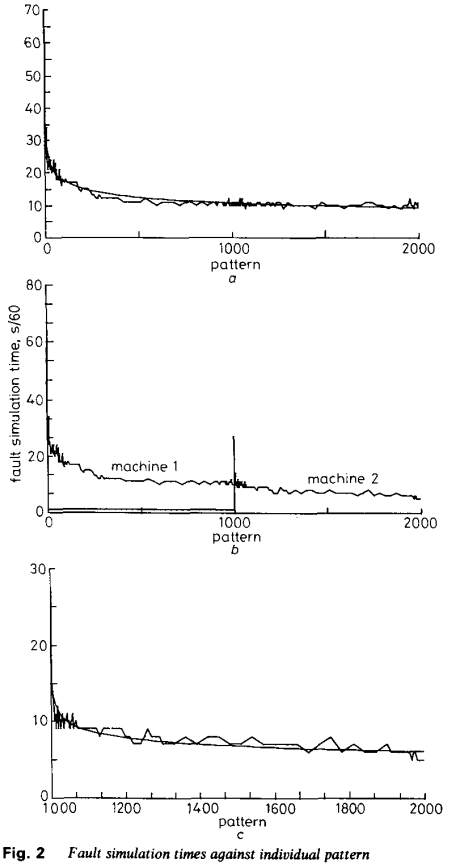


Fig. 2 *Fault simulation times against individual pattern*

a One-machine simulation for circuit s5378 for 2000 random patterns; curve is $y = 46.9\,x^{-0.214}$, where y is simulation time and x is pattern number
b Similar curves but for 2-machine simulation without communication
c Expanded curve of second machine simulation of (*b*); curve is $y = 17.8\,x^{-0.156}$

speedup function is

$$S_n = \frac{T_1}{T_{1_n}} = \frac{\alpha}{\alpha_n}\frac{1-r_n}{1-r}P^{r_n-r}n^{1-r_n} \tag{5}$$

Since α is the time spent on simulating the first pattern and it is proportional to the number of total faults, all the α's of the distributed fault simulations are the same. That is

$$\alpha = \alpha_2 = \cdots = \alpha_n \tag{6}$$

Also, at the end of each distributed fault simulation, all the remaining faults are hard-to-detect faults. These faults are the same for each distributed fault simulation, and the time spent for simulating them will be the same for each simulation. That is

$$T_s(P) = T_{s_2}(P/2) = \cdots = T_{s_n}(P/n) \tag{7}$$

The relations among the decay indices can be obtained by combining eqns. 3, 6 and 7

$$P^{-r} = (P/2)^{-r_2} = \cdots = (P/n)^{-r_n} \tag{8}$$

The factor $P^{r_n-r}n^{1-r_n}$ in eqn. 5 equals to n when eqn. 8 is substituted into it. So the efficiency of this distributed fault simulation S_n/n can be written as

$$\frac{S_n}{n} = \frac{(1-r)\ln P - \ln n}{(1-r)(\ln P - \ln n)}$$

$$= 1 - \frac{r \ln n}{(1-r)(\ln P - \ln n)} \tag{9}$$

4 Experimental results and discussions

The distributed fault simulation by pattern partitioning (DFSPP) scheme for sequential circuits described has been implemented in C language on Sun4/SPARC2 workstations. The workstations were connected with the Ethernet for which the data transfer rate is 30–40 Kbyte/s. The ISCAS'89 sequential circuits [25] were simulated.

Table 1 shows the statistics of the distributed fault simulation for simulating 2000 random patterns and

Table 1: Speedup of DFSPP

Circuits	No. of pat.	Fault_ cov. (%)	Sim. time (s)	Comm. time (s)	Machines 2	3	4	5
Random patterns								
s382	2000	12.08	47	0.09	2.05	3.09	3.99	4.80
s400	2000	12.11	50	0.09	2.11	3.20	4.15	4.93
s420	2000	25.94	48	0.10	4.02	5.26	6.80	7.60
s444	2000	10.55	57	0.09	1.93	2.80	3.60	4.30
s526	2000	8.18	76	0.09	1.99	2.90	3.77	4.59
s820	2000	45.24	38	0.14	1.95	2.84	3.70	4.41
s832	2000	44.30	39	0.14	1.95	2.89	3.76	4.49
s838	2000	21.56	125	0.11	4.42	6.32	7.93	8.96
s953	2000	7.54	541	0.10	1.95	2.92	3.87	4.75
s1423	2000	41.76	165	0.17	2.41	3.32	4.31	5.13
s1488	2000	56.78	76	0.21	2.61	3.42	4.20	4.97
s1494	2000	55.82	77	0.21	2.63	3.47	4.23	5.00
s5378	2000	65.11	359	0.51	2.50	3.68	4.46	5.09
s9234	2000	0.37	768	0.09	1.99	2.98	4.00	4.80
s13207	2000	6.74	4228	0.17	2.10	3.01	4.01	4.91
s15850	2000	3.54	5983	0.13	2.09	3.10	3.93	4.82
s35932	2000	74.56	2111	3.91	2.45	3.54	4.38	5.11
s38417	2000	3.21	17081	0.23	2.02	3.16	3.98	4.96
s38584	2000	34.24	23467	1.87	2.19	3.23	4.00	4.88
ATPG patterns								
s382	2463	89.89	74	0.15	3.36	4.58	6.04	7.28
s400	1282	81.84	44	0.10	2.60	4.11	4.95	5.79
s444	1881	88.01	80	0.13	3.10	4.32	5.97	6.95
s1488	590	92.54	39	0.24	2.99	3.88	4.51	5.53
s1494	469	91.03	39	0.23	2.99	3.91	4.54	5.58
s5378	408	74.26	82	0.52	2.87	3.68	4.49	5.34
s35932	86	86.86	205	5.65	2.27	3.13	4.01	4.84

ATPG patterns. In the Table, the simulation times are fault simulation times for one-machine and the communication times are communication times spent in transmitting detected faults for five-machine simulation. It can be seen that the communication time between machines during the simulation is small as compared with the total simulation time. The speedup is equal to the fault simulation times for 1 machine divided by the total simulation times, including the fault-free simulation times and the communication times, for n machines. The speedup ratio increases with the number of machines, and the speedup can exceed the number of machines for some circuits. That is, the distributed fault simulation exhibits a superlinear speedup. For the ATPG patterns, this superlinear

speedup is more significant. The reason for this superlinear speedup is caused by the Case 1 faults, which is discussed later, for the distributed pattern-partitioning fault simulation.

In sequential fault simulations, there are some cases of faults which affect the simulation time significantly. They are discussed as follows.

Case 1: This case is caused by the fault effect which makes the D-type flip-flop (DFF) be uninitialised. The stuck_at_1 fault on line B, indicated by the arrow in the circuit example in Fig. 3 is such a case. In Fig. 3*a*, the

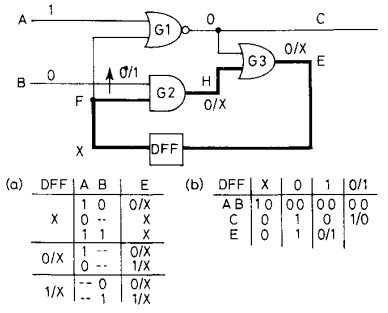


Fig. 3 *Example to explain case 1 faults*

a If DFF is initially set at unknown state the fault B/1 is untestable
b Fault B/1 can be dropped if fault-free simulation is performed first

DFF is initially at X and the value on E will be either $0/X$ or X. If the value on DFF is $0/X$, or $1/X$, the value on E will be either $0/X$ or $1/X$. It can be seen that no matter what sequence patterns are applied to (A, B), the DFF is always at X under the existence of fault B/1 and the fault effect circles around the loop G2, G3, and DFF. This wastes the simulation time. However, if only the fault-free simulation, i.e. no fault is injected, is performed for the patterns (A, B) = (1, 0) to firstly set the DFF to be the state '0' (Fig. 3*b*), the fault B/1 can be activated and propagated to output C by the following three-pattern sequence (A, B) = (0, 0), (0, 0), and (0, 0). Furthermore, the fault is considered to be untestable in the normal fault simulation, however, it is truly detectable no matter which initial value (0 or 1) is on DFF when the previous pattern sequence (1, 0), (0, 0), (0, 0), and (0, 0) is applied to the circuit.

The numbers of the Case 1 faults for simulating 2000 random patterns and ATPG patterns are listed in Table 2. From the Table, it can be seen that the numbers of the Case 1 faults are generally small as compared to the total faults (less than 1.0% in general). Although, the Case 1 faults occupy only a small portion of the total faults, they consume so much of the simulation time that DFSPP, after dropping them, gets the superlinear speedup in Table 1.

The Case 1 faults were further studied to see how they affect the fault simulation. Fig. 4 shows the plots of the elapsed simulation times against the pattern for s5378 for four simulations, where True_i means that the simulation is first performed fault-free simulation with i patterns and True_0 is the normal fault simulation. It can be seen that the elapsed time for curves True_2 and True_3 is approximately one half of that of the curves True_0. It means that all the Case 1 faults for True_2 and True_3 simulations had been dropped and the time consumed for these faults is about one half of the fault simulation time in simulating the large number of patterns. Also it is observed that curves True_2 and True_3 are always

Table 2: Numbers of case 1 faults, ODFs and UDFs

Circuits	Faults	Case 1 fault				ODF				UDF			
		Machines				Machines				Machines			
		2	3	4	5	2	3	4	5	2	3	4	5
Random patterns													
s382	356	1	1	1	1	0	0	0	0	0	0	0	0
s400	380	1	1	1	1	0	0	0	0	0	0	0	0
s420	397	25	25	25	25	0	0	0	0	0	0	0	0
s444	417	1	1	1	1	0	0	0	0	0	0	0	0
s526	513	1	1	1	1	0	0	0	0	0	0	0	0
s820	840	0	0	0	0	0	0	0	0	0	0	0	0
s832	860	0	0	0	0	0	0	0	0	0	0	0	0
s838	793	49	49	49	49	0	0	0	0	0	0	0	0
s953	1021	0	0	0	0	0	0	0	0	0	0	0	0
s1423	1365	15	14	16	17	1	1	1	1	0	0	0	4
s1488	1474	1	1	1	1	0	0	0	0	0	0	0	0
s1494	1494	1	1	1	1	0	0	0	0	0	0	0	0
s5378	4239	12	12	12	12	0	0	0	0	0	0	0	0
s9234	2672	0	0	0	0	0	0	0	0	0	0	0	0
s13207	7540	35	36	35	35	0	0	0	0	0	0	0	0
s15850	7768	12	12	12	12	0	0	0	0	0	0	0	0
s35932	33243	10	10	10	10	0	0	0	0	0	0	0	0
s38417	26970	25	25	25	24	0	0	0	0	0	0	0	0
s38584	32815	93	92	106	97	7	6	8	6	8	10	25	16
ATPG patterns													
s382	356	13	13	13	13	0	0	0	0	0	0	0	0
s400	380	14	14	14	14	0	0	0	0	0	0	0	0
s444	417	12	12	12	12	0	0	0	0	0	0	0	0
s1488	1474	2	2	2	2	0	0	0	0	0	0	0	0
s1494	1494	4	4	4	4	0	0	0	0	0	0	0	0
s5378	4239	18	18	18	18	0	0	0	0	0	0	0	0
s35932	33243	10	10	10	10	0	0	0	0	153	428	493	462

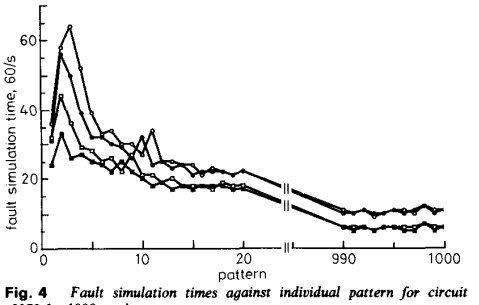


Fig. 4 *Fault simulation times against individual pattern for circuit s5378 for 1000 random patterns*

Curves True_0, True_1, True_2 and True_3 means that zero, one, two and three initial patterns were first performed in fault-free simulation

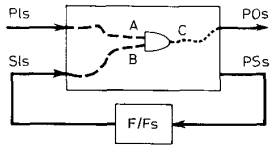
	total time	faults remaining
—○— True_0	223	1527
—●— True_1	221	1527
—□— True_2	144	1515
—■— True_3	138	1515

lower than the curve True_0 for the initial pattern simulation. This indicates that the Case 1 faults had been detected in the first several patterns for the True_2 and True_3 simulations. When the total simulation times are compared, the difference in simulation times between True_3 and True_0 is 85 seconds which is 35% of the total simulation time, while the number of the Case 1 faults is 12, which is only 0.35% of the total faults. For the n-machine system, the fault simulation of the first machine is True_0 simulation and those of the other machines are analogies of True_3 simulation. The Case 1 faults cannot be detected by the simulation done the first machine but can be detected at the early stage of simula-

tion done the other machines. Through communication on a network, the Case 1 faults can also be dropped early for the first machine, and thus DFSPP can get super-linear speedup.

A fault is a potentially detectable fault (PDF) if the fault is sequentially undetectable and there is an input sequence that produces a combination of the good and faulty output responses $0/X$ or $1/X$ at some primary outputs if every DFF starts at the unknown state [26]. From the previous discussion, the Case 1 faults are PDFs. It has been shown that potentially undetectable faults can be identified by using a test pattern generation process with a four-valued logic: 0, 1, X (don't care) and U (unknown) [26]. That process needed to try every choice in the test generation and consumed much time. However, with the DFSPP, the Case 1 faults can be identified and dropped in the first several patterns. It pays no extra penalty.

Case 2: This case is caused by the sequential self-masking effect [27]. Figs. 5*a* and 5*b* is used to demon-



	A	B	C	
(a)	0/1	1/0	0	
(b)	0/1	1	**0/1**	**ODF**
(c)	0/1	0/1	0/1	
(d)	0/1	**0**	**0**	**UDF**

Fig. 5 *Example to explain ODF and UDF*
a and *c* Original fault simulation
b and *d* Fault simulation by pattern partitioning

strate this, From the original fault simulation as shown in Fig. 5*a*, the value on A is 1/0 which is the fault effect of a fault in the previous time frame, the value on B is 0/1 which is the fault effect of the same fault in the present time frame. The fault cannot be detected because of the self-masking effect. However, for the pattern partitioning fault simulation as shown in Fig. 5*b*, the value on B was set to 1 as in the previous example. The fault will be propagated to POs and be detected because of disappearance of the self-masking effect. We define the Case 2 faults as over-detected-faults (ODFs), which are detected by DFSPP but not detected by the normal fault simulation.

Case 3: The faults of this case are caused by the sequential multipath sensitisation effect [27]. Figs. 5*c* and 5*d* is used to explain this effect. In Fig. 5*c*, the fault effect (0/1) on A can be propagated to POs under the original fault simulation. However, for the pattern-partitioning fault simulation shown in Fig. 5*d*, the fault effect is blocked by the value 0, which was set by a fault-free simulation sequence. We define the Case 3 faults as under-detected faults (UDFs), which can be detected by the original fault simulation but become undetected faults for DFSPP.

The numbers of ODFs and UDFs are also listed in Table 2. It can be seen that only few ODFs and UDFs occur for s1423, s35932 and s38584.

In Section 3, we formulated the speedup of DFSPP from the theoretical point of view. Fig. 6 shows the plots of the speedups against the numbers of machines of the

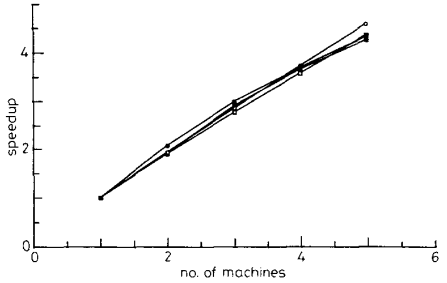


Fig. 6 *Plots of speedup against number of machines of theoretical estimates and experimental results for circuits s5378 and s35932*
—□— estimate } s5378
—■— experiment }
—○— estimate } s35932
—●— experiment }

theoretical estimations and the actual experimental results, which were run with first removing the case 1 faults, with simulating 2000 random patterns for circuits s5378 and s35932, for which the decay indices were 0.32 and 0.23, respectively. It can be seen that two sets of the curves match closely even for the large circuit s35932. Fig. 7 shows the plots of the speedups against the

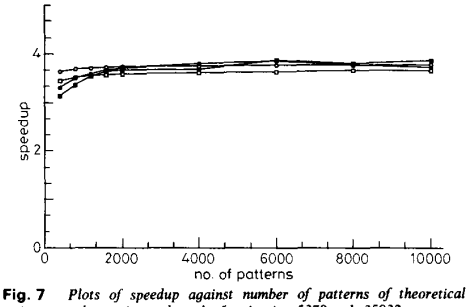


Fig. 7 *Plots of speedup against number of patterns of theoretical estimates and experimental results for circuits s5378 and s35932*

numbers of patterns with four-machine simulation. It can be seen that the experimental speedups are lower than those obtained from the theoretical estimation for the small numbers of patterns because the communication time for transmitting detected faults is neglected in the estimation. For the large number of patterns, the communication times are small when compared with the total simulation times and the two speedup curves match closely.

4 Conclusion

We have proposed a distributed fault simulation by pattern partitioning for sequential circuits in a distributed network of workstations. For this distributed fault simulation, the number of patterns needed to perform fault simulation for each machine is reduced by a factor of n, the number of machines, and the faults detected by any machine can be dropped through communication of the network. The communication time can be neglected

since only detected faults need to be transmitted and this takes negligible time as compared with the total fault simulation time. The distributed fault simulation can achieve a superlinear speedup by automatically dropping the case 1 faults, which usually cause much simulation time and would be considered to be undetected faults in the traditional three-valued fault simulation but are truly detected in fact. Almost 'exact' results can be obtained on the detected faults except for a small percentage of ODFs and UDFs, which occupy only 0.3% of the total faults for the worst case. A mathematical model has also been presented to predict the performance of this pattern-partitioning distributed fault simulation.

5 References

1 SESHU, S.: 'On an improved diagnosis program', *IEEE Trans.*, 1965, **EC-12,** (2), pp. 76–79

2 ARMSTRONG, D.B.: 'A deductive method of simulatintg faults in logic circuits', *IEEE Trans.*, 1972, **C-21,** (5), pp. 464–471

3 ULRICH, E.G., and BAKER, T.G.: 'Concurrent simulation of nearly identical digital networks', *Computer*, 1974, **7,** (4), pp. 39–44

4 ARAMOVICI, M., MENON, P.R., and MILLER, D.T.: 'Critical path tracing: an alternative to fault simulation', *IEEE Des. Test Comput.*, 1984, **1,** (1), pp. 83–93

5 NIERMANN, T.M., CHENG, W.T., and DATAL, J.H.: 'PROOFS: a fast, memory efficient sequential circuit fault simulator', *IEEE Trans.*, 1992, **CAD-11,** (2), pp. 198–207

6 GOUDERS, N., and KAIBEL, R.: 'PARIS: a parallel pattern fault simulator for synchronous sequential circuits'. Proceeding of international conference on *Computer aided design*, 1991, pp. 542–545

7 MOJTAHEDI, M., and GEISSELHARDT, W.: 'New methods for parallel pattern fast fault simulation for synchronous sequential circuits'. Proceeding of international conference on *Computer aided design*, 1993, pp. 2–5

8 LEE, H.K., and HA, D.S.: 'HOPE: an efficient parallel fault simulator for synchronous sequential circuits'. Proceedings of 29th conference on *Design automation*, 1992, pp. 336–340

9 LEE, H.K., and HA, D.S.: 'New methods of improving parallel fault simulator in synchronous sequential circuits'. Proceedings of international conference on *Computer aided design*, 1993, pp. 10–17

10 WU, C.P., LEE, C.L., and SHEN, W.Z.: 'SEESIM — a fast synchronous sequential circuit fault simulator with single event equivalence'. European conference on *Design automation*, 1992, pp. 446–449

11 PFISTER, G.F.: 'The Yorktown simulation engine: Introduction'. Proceedings of 19th conference *Design automation*, 1982, pp. 51–54

12 SASAKI, T., KOIKE, N., and TOMITA, K.: 'HAL: a block level hardware logic simulator'. Proceedings of 20th conference on *Design automation*, 1983, pp. 150–156

13 BLANK, T.: 'A survey of hardware accelerator used in computer-aided design', *IEEE Des. Test Comput.*, 1984, **1,** (4), pp. 21–39

14 AGRAWAL, P., DALLY, W.J., FISCHER, W.C., JAGADISH, H.V., KRISHNAKUMAR, A.S., and TUTUNDJIAN, R.: 'MARS: a multiprocessor-based programmable accelerator', *IEEE Des. Test Comput.*, 1987, **4,** (5), pp. 28–36

15 AGRAWAL, P., AGEAWAL, V.D., and CHENG, K.T.: 'Fault simulation in a pipelined multiprocessor system'. Proceedings of international conference on *Test*, 1989, pp. 727–734

16 PATIL, S., BANERJEE, P., and PATEL, J.H.: 'Parallel test generation for sequential circuits on general-purpose multiprocessors'. Proceedings of 28th conference on *Design automation*, 1991, pp. 155–159

17 MUELLER-THUNS, R.B., SAAB, D.G., DAMIANO, R.F., and ABRAHAM, J.A.: 'Portable parallel logic and fault simulation'. Proceedings of international conference on *Computer aided design*, 1989, pp. 506–509

18 DUBA, P.A., ROY, R.K., ABRAHAM, J.A., and ROGERS, W.A.: 'Fault simulation in a distributed environment'. Proceedings of 25th conference on *Design automation*, 1988, pp. 686–691

19 PATIL, S., and BANERJEE, P.: 'Fault partitioning issues in an integrated parallel test generation/fault simulation environment'. Proceedings of international conference on *Test*, 1989, pp. 718–726

20 PATIL, S., and BANERJEE, P.: 'Performance trade-offs in a parallel test generation/fault simulation environment'. *IEEE Trans.*, 1991, **CAD-10,** (12), pp. 1542–1558

21 AGUADO, M.J., DE LA TORRE, E., MIRANDA, M.A., and LOPEZ-BARRIO, C.: 'Distributed implementation of an ATPG system using dynamic fault allocation'. Proceedings of international conference on *Test*, 1993, pp. 409–418

22 WARSHAWSKY, A., and RAJSKI, J.: 'Distributed fault simulation with vector set partitioning'. Proceedings of European conference on *Test*, 1990, pp. 227–236

23 MARKAS, T., ROYALS, M., and KANOPOULOS, N.: 'On distributed fault simulation', *Computer*, 1990, **23,** (1), pp. 40–52

24 FUJIWARA, H., and INOUE, T.: 'Optimal granularity of test generation in a distributed system', *IEEE Trans.*, 1990, **CAD-9,** (8), pp. 885–892

25 BRALEZ, F., BRALEZ, D., and KOZMINSKI, K.: 'Combinational profiles of sequential benchmark circuits'. Proceedings of international symposium on *Circuits and systems*, 1989, pp. 1929–1934

26 CHENG, K.T.: 'On removing redundancy in sequential circuits'. Proceedings of 28th conference on *Design automation*, 1991, pp. 164–169

27 CHEN, J.E., LEE, C.L., and SHEN, W.Z.: 'Single-fault fault-collapsing analysis in sequential logic circuits', *IEEE Trans.*, 1991, **CAD-10,** (12), pp. 1559–1568