SEESIM: a fast synchronous sequential circuit fault simulator with single-event equivalence

C.P. Wu
C.L. Lee
W.Z. Shen

Indexing terms: Fault simulation, Single-event equivalence

Abstract: The paper presents a concept of single event equivalence to be used in the sequential circuit fault simulator. The concept dynamically identifies the equivalent faults for a simulated pattern. It combines advantages of the fanout-free region, critical path tracing and the dominator concept, which were applicable only to combinational circuit fault simulation. The implemented fault simulator, SEESIM, based on the concept, demonstrated a performance superior to that of a state-of-the-art concurrent fault simulator, and comparable to that of parallel-pattern single-fault propagation simulators. It requires a minimal amount of memory and, because of its simplicity, can be easily extended to multilogic or higher level simulation.

1 Introduction

A fault simulator is used to simulate faults and to evaluate the fault coverage for test patterns. Many schemes had been employed to improve the speed of fault simulators. The classical schemes include parallel fault simulation [1], deductive fault simulation [2] and concurrent fault simulation [3]. With the scan design methods [4–7] currently in use, the parallel-pattern single-fault propagation (PPSFP) fault simulator for combinational circuits [8] has gained much attention. Many techniques addressing how to improve the fault simulation speed of PPSFP for combinational circuits, such as critical path tracing, the fanout-free region, the stem region [9] and the dominator concept [10], have been proposed. Unfortunately, none of these can be applied to sequential circuits [11]. A parallel fault simulator PROOFS [11], which was evolved from DSIM [12], can handle sequential circuits. However, the parallel fault scheme used in PROOFS makes it sensitive to the order of the faults simulated [11]. Also, the parallel fault is inefficient when extended to multivalues (more than 0, 1, X, Z) and higher level simulation because a coding scheme needs to be used.

In this paper, a concept of single-event equivalence (SEE) is proposed and demonstrated. It combines the advantages of the fanout-free region [13], critical path tracing [14] and the dominator concept [10], which were applicable only to combinational simulation, in sequential simulation. A simulator was implemented to demonstrate the effectiveness of this approach. Experimental results show that the simulator has a performance superior to that of a state-of-the-art commercial concurrent fault simulator and comparable to that of PPSFP simulation. It requires a minimal amount of memory: on average, only 172 bytes to process a gate. This means that it is possible to simulate a circuit of 100000 gates with only 18 Mbytes of main memory with this simulator.

© IEE, 1993

Paper 9246G (E1, E10), first received 11th November 1991 and in revised form 20th July 1992

The authors are with the Department of Electronics Engineering & Institute of Electronics, National Chiao Tung University, Hsin-Chu, Taiwan, ROC

2 Concept of single-event equivalence

2.1 Equivalent fault collapsing

The synchronous sequential circuit model considered is shown in Fig. 1, where $x_1, \ldots, x_m$ and $z_1, \ldots, z_n$ are the primary inputs and the primary outputs of the sequential

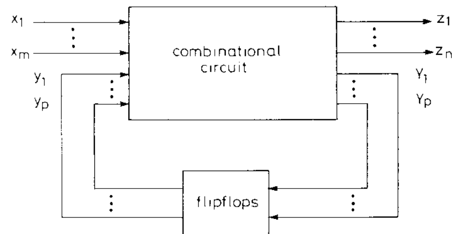

Fig. 1 *Model of a synchronous sequential circuit*

circuit, respectively. $Y_1, \ldots, Y_p$ and $y_1, \ldots, y_p$ are the next and the present state variables of the sequential circuit, respectively. $Y_1, \ldots, Y_p, z_1, \ldots, z_n$ and $x_1, \ldots, x_m, y_1, \ldots, y_p$ are the outputs and inputs, respectively, of the combinational part of the circuit.

For the synchronous sequential circuit of Fig. 1 with two faults, f1 and f2, we have the following definitions.

Definition 1: f1 and f2 are *combinationally equivalent* if the presence of f1 or f2 results in the same response on

The authors are grateful to Dr. K.T. Cheng for his supply of the test patterns for the ISCAS'89 benchmark circuits, and to Miss Jane Wang for her help in running the state-of-the-art commercial fault simulator. The financial support from National Science Council of the Republic of China through the contract NSC-80-0404-E-009-34 is also acknowledged.

the outputs of the combinational circuit when the same patterns are applied on the inputs of the combinational circuit.

Definition 2: f1 and f2 are *partially equivalent* for the test sequence T if the presence of f1 or f2 results in the same response in the outputs of the combinational part of the circuit, on applying T to the primary inputs, when two faulty machines are initially in the same state.

Definition 3: f1 and f2 are *sequentially equivalent* if the presence of f1 or f2 results in the same response in the primary outputs when two faulty machines are initially in the same state and the primary inputs are subjected to the same test sequence.

With the above definitions, we have the following lemma.

Lemma 1: Faults that are combinationally equivalent are sequentially equivalent.

Proof: This lemma can be proved by contradiction. Let two faults, f1 and f2, be combinationally equivalent but not sequentially equivalent. There exists at least one test sequence T_D which can distinguish the fault effects of these two faults at primary outputs of the circuit. This implies that, on applying the input sequence T_D to the primary inputs of the circuit, there exists a pattern of inputs to the combinational part of the circuit that produces different values in outputs of the combinational part of the circuit. This conflicts with the definition of combinational equivalence. Hence, f1 and f2 must be sequentially equivalent.

Definition 4: For a fault-free circuit, line A is the *controller* of line B if setting line A to the logic value v_1 results in line B being at the logic value v_2 for any pattern of inputs to the combinational circuit; v_1 is called the *controlling value* and v_2 the *controlled value.*

Definition 5: Line B is the *dominator* of line A if all the paths from line A to the primary outputs pass through line B.

The above definition is the same as that given in Reference 10.

Lemma 2: For combinational circuits, if a pattern T detects a fault at line A, it also detects a fault at line B if line B is the dominator of line A.

Proof: The proof is trivial because the sensitised path under the pattern T must pass through line B.

Theorem 1: For two lines A and B in a sequential circuit, if (1) B is the dominator of A and (2) line A at the value v_1 is the controller of line B at the value v_2, then faults A-sa-v_1 and B-sa-v_2 are sequentially equivalent.

Proof: From Lemma 1, we have only to prove that A-sa-v_1 and B-sa-v_2 are combinationally equivalent. The theorem is also to be proved by contradiction. Assume that both faults have different responses at outputs of the combinational circuit under the pattern $T = \{x_1, \ldots, x_m, y_1, \ldots, y_p\}$, which is applied to the inputs of the combinational circuit. For each output of the combinational circuit, there exist two cases that can distinguish the fault effects of B-sa-v_2 and A-sa-v_1: first, B-sa-v_2 is detected at the output but A-sa-v_1 is not and, second, A-sa-v_1 is detected at the output but B-sa-v_2 is not. For the first case, the detection of B-sa-v_2 implies that the fault-free value of line B is $'v_2$ (the complement of v_2) and that of line A is $'v_1$ from Definition 4. From Condition 2, the fault effect of A-sa-v_1 also appears at line B and is detected at the output as the fault effect of B-sa-v_2. This conflicts with the assumption. For the second case, B-sa-v_2 must be detected from Condition 1 and Lemma 2. This also conflicts with the assumption. Therefore, the A-sa-v_1 and B-sa-v_2 must be sequentially equivalent and the theorem is proved.

Based on Theorem 1, we derive a procedure as follows to find the sequentially equivalent faults that satisfy Conditions 1 and 2 in the theorem. This procedure examines every fault one by one to check whether the fault is equivalent to another fault. In examining a fault, there are two kinds of event that need to be considered: the path event for finding the line that satisfies Condition 1, and the logic event for verifying the lines that satisfy Condition 2.

Fig. 2 demonstrates how to check whether the fault is equivalent to another fault. In the figure, the circuit has

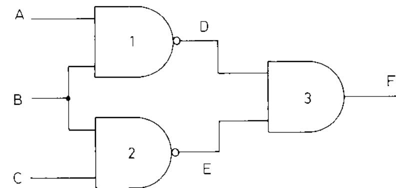

Fig. 2 *An example to demonstrate how to find equivalent faults*

been levelled from inputs to outputs, and initially all lines of the circuit are at the unknown value X. To examine the fault equivalence for the fault B-sa-0, a path event is activated on line B. This will propagate to lines D and E to become two events. An event queue can be used to store these events. Next, to further propagate the events along the circuit, event D is taken from the queue to be processed. This event will propagate to line F, and F will be stored in the event queue. Next, event line E will be taken out and processed. It will also propagate to line F. Because event line F has already been in the event queue, there is only a single event in the event queue, and it is then known that line F is the dominator of line B. At this time, all line events originating from line B converge to line F to be a single line event. The above is the path event evaluation. Also, during evaluation of the path events, logic events are processed to evaluate the logic value on each line caused by fault B-sa-0. In this example, the lines A, B, C, D, E and F are evaluated in sequence to have values X, 0, X, 1, 1 and 1, respectively. If a line is found to be a dominator of the faulty line during the path event evaluation, the logic value of this line is checked. If it is not X, the fault at this line with this logic value is equivalent to the fault of the faulty line. In this example, the fault B-sa-0 is equivalent to F-sa-1.

In the above example, the path event evaluation checks whether paths converge to a single line, i.e. it checks for Condition 1 of Theorem 1, and the logic event evaluation checks the controlling and controlled relationship of the dominating and dominated lines, i.e. it checks for Condition 2 of Theorem 1. If Condition 1 is met, it is said that SEE exists for the fault being checked. This SEE checking can be expanded to the pattern-dependent case and sequential circuits; it serves as the foundation for this simulator.

A formal description of the above procedure to find the equivalent fault is listed below. For this procedure, the combinational part of the circuit has been levelled from combinational inputs to combinational outputs. In the description, each fault is taken from the *fault_list* and is examined in the **for** loop. The *event_number* is used to record the number of events in the event queue to be processed and the *last_value* is the faulty value of the last examined line.

Procedure 1: (sequential equivalent fault collapsing)

```
set logical values of all lines to X;
for (each activated fault in the fault_list){
 inject fault;
 while (event_number is not 0){
  gate = GET_EVENT( ); /* get event: the lower level first */
  last_value = EVALUATE(gate);
  if (the value of last_value is not X){
   if (event_number is 0){

    /* an equivalent fault is found */

    drop fault from fault_list;
    break; /* stop the while loop */
   }
  }
  for (each fan_out of gate){
   if (fan_out is a combinational output){
   event_number = 0; /* used to stop the while loop */
    break; /* stop this for loop */
   } else {
    EVENT_SCHEDULING(fan_out);
   }
  }
 }
 Recover every changes made by evaluating of this fault;
}
```

In the above procedure, the events originating from a fanout stem do not necessarily converge to one dominator line. Events may propagate to combinational outputs of the circuit. For this case, no equivalence relationship exists and the procedure stops.

2.2 Pattern-dependent single-event equivalence

In the previous Section, SEE was applied by considering only the static structure of the circuit. No pattern was applied during checking of the SEE. When a pattern is applied, SEE between faults can still be checked. However, at this time, logic values at all circuit lines are fixed. The equivalences between faults are pattern dependent. Procedure 1 can be modified as follows to check the equivalence between faults.

Procedure 2: (pattern dependence SEE checking)

```
For a simulated pattern on inputs:
do true simulation for the circuit.
for (each undetected fault in the fault_list){
 inject fault and recover the faulty state of this fault;
 while (event_number is not 0){
  gate = GET_EVENT( ); /* get event: the lower level first */
  last_value = EVALUATE(gate);
  if (the last_value differs from the fault free value of gate){
   if (event_number is 0){

    /* SEE condition is identified */

    break; /* stop the while loop */
   }
   for (each fan_out of gate){
    if (fan_out is a combinational output){
    event_number = 0; /* used to stop the while loop */
     break; /* stop this for loop */
    }else{
     EVENT_SCHEDULING(fan_out);
    }
   }
  }
 }
 Recover every changes made by evaluating of this fault;
}
```

For the example circuit of Fig. 2, if the applied pattern is (A, B, C) = 1, 0, 1), the application of Procedure 2 shows first that (A, B, C, D, E, F) = (1, 0, 1, 1, 1, 1). For the injected fault B-sa-1, the initiated fault event B-sa-1 will propagate to line D and E. Similarly to Procedure 1, these two events will propagate to line F to become a single event and it can be found that fault F-sa-0 is equivalent to B-sa-1.

In the above example, the SEE property can be used not only to identify the equivalence relationship due to the existence of dynamic domination between faults, but also to identify the equivalence relationship between faults of a far more general case. For example, in the circuit example of Fig. 2, if the applied pattern is (A, B, C) = (1, 1, 0), fault A-sa-0 is quickly found to be equivalent to fault D-sa-1, because the fault event initiated at line A propagates to line D with the single-event criteria satisfied. In a similar way, for the same pattern, it is found that B-sa-0 is equivalent to D-sa-1, and D-sa-1 to F-sa-1. That is, for this pattern, faults A-sa-0, B-sa-0, D-sa-1 and F-sa-1 are detected.

In the above example, faults B-sa-1 and F-sa-0 are partially equivalent under the pattern (A, B, C) = (1, 0, 1), and faults A-sa-0, B-sa-0, D-sa-1 and F-sa-1 are partially equivalent under the pattern (A, B, C) = (1, 1, 0). It is to be expected that the number of faults to have the partially equivalent relationship will be larger than the number of faults to have sequential equivalence because the conditions for sequential equivalence are more stringent.

Pattern-dependent SEE can be applied to sequential circuits. For example, Fig. 3 shows an iterative expansion

Fig. 3 *An example to demonstrate how single-event equivalence works in a sequential circuit*

of a part of a circuit for time frames N and $N + 1$ to find the fault equivalence for the fault B-sa-1. The patterns simulated for these two time frames are (A, B, C) = (1, 0, 1; 1, 0, 0), and for the $(N - 1)$th time frame the output of the D flip-flop is 0. From Procedure 2, line B-sa-1 is equivalent to D-sa-1 because there is an SEE condition on line D for the time frame N. For the $(N + 1)$th time frame, the SEE condition exists on line E after the events from B and D propagate to line E, which is a primary

output. Line B-sa-1 is equivalent to E-sa-1. As the fault E-sa-1 is detectable under this pattern set, so is B-sa-1.

It is possible to compare the above pattern-dependent SEE technique with the fanout-free region technique [13], which has often been used in fault simulation for combinational circuits. In the SEE technique, the logical cone is automatically identified during the process of fault simulation. (The logical cone for the primary output E is indicated in Fig. 3.) Moreover, it is larger than the static structural fanout-free region and can cross the boundaries of time frames. Hence, the number of equivalent faults identified will be larger.

It is also possible to compare the SEE technique with the critical path tracing technique, which is often used [9, 10] to identify the faults detected for a simulated pattern. The technique is efficient and fast, although it is only an approximate method and gives a pessimistic result for fault coverage [14]. The SEE technique essentially traces all the sensitised paths in a levelled order from primary inputs to primary outputs. It is much simpler and needs a smaller overhead in memory usage. More importantly, it is an exact method.

Finally, pattern-dependent SEE expands the structural domination relationship in the previous section into a dynamic domination relationship. For example, in the circuit in Fig. 4, structurally, line N is not a dominator of line M, because the paths from M to the primary outputs can pass through either P or Q instead of N. However, if L1 and L2 have the value 0, line N will become the dominator for line M, because all the sensitised paths from M to the primary outputs pass through N with this pattern.

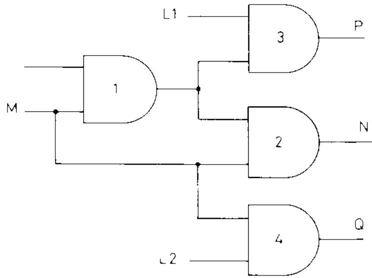

Fig. 4 *An example to demonstrate that single-event equivalence automatically identifies the logical domination relationship*

3 Experimental results

The simulator SEESIM was implemented in C language under the UNIX system. It had been applied to ISCAS benchmark circuits [15, 16] to verify its performance. The simulation times are the CPU time in seconds of a SUN 4/260 workstation.

To investigate the effect of SEE in saving simulation time, another fault simulator which was very similar to SEESIM in implementation but did not employ the SEE scheme was also implemented. SEESIM and this simulator were applied to ISCAS'85 combinational benchmark circuits [15] to simulate faults for 200 random patterns. During simulation, the numbers of event evaluations and the simulation times were recorded and compared. Table 1 shows the results. The level of fault coverage for each circuit is also included. It can be seen that the mean savings on event evaluation are approximately 43% and the saving on computation times are 30%. It can also be seen that the savings are circuit dependent. In general, the larger the circuit size and level, the larger the savings that can be obtained. This is obvious because, for a circuit of larger size and level, more SEE relationships will be found between faults.

Table 1: Savings on event evaluations and computation times with the single-event equivalence scheme to simulate ISCAS'85 benchmark circuits for 200 random patterns

Circuit	Level	Fault coverage (%)	Savings (%)	
			Event evaluation	Time
c432	18	92.7	14.1	6.3
c499	12	95.4	50.6	31.2
c880	25	94.5	22.8	12.8
c1355	25	89.9	67.4	47.6
c1908	41	81.5	51.0	43.0
c2670	33	80.7	34.0	25.1
c3540	48	87.4	34.2	25.2
c5315	50	96.8	49.9	30.8
c6288	12	99.6	63.9	50.5
c7552	44	89.3	45.8	32.2
Mean	30.8	90.8	43.4	30.4

The simulation times of SEESIM can be compared with those of PPSFP simulators [9, 10] as shown in Table 2. The number of simulated patterns is 224. The simulation times in Table 2 have been normalised to SEESIM by taking into account the speeds of the machines. The SEESIM simulation times are comparable to those of PPSFP simulators even though no parallelism was employed in SEESIM for its true value simulation.

Table 2: Normalised simulation times of PPFSP simulators compared with SEESIM

Circuit	SEESIM	Dominator	Tulip
c432	1	0.97	2.59
c499	1	1.64	1.52
c880	1	1.18	1.53
c1355	1	1.65	1.68
c1908	1	0.73	1.47
c2670	1	0.63	1.09
c3540	1	0.88	2.93
c5315	1	0.89	1.77
c6288	1	3.88	n.a.
c7552	1	0.69	2.23
Mean	1	1.31	1.87

SEESIM was also compared with a state-of-the-art commercial concurrent fault simulator (CFS) on running 27 ISCAS'89 sequential benchmark circuits [16]. Table 3 lists the numbers of simulated patterns, the fault coverages, the simulation times (in seconds) and the memory spent on each circuit for the CFS and SEESIM. The ratios of the time spent and the memory usage for the CFS to SEESIM for each circuit are also included. The test patterns were randomly generated. Circuit s420 had been modified because parts of the original circuit could not be observed or controlled. Circuits larger than s35932 were too large to be simulated by the CFS on a 32 Mbyte machine. For all circuits, SEESIM exhibited a superior performance over that of CFS. It was approximately five times faster than CFS. The reduction ratio in the memory usage is approximately 12. The mean size of memory required for SEESIM to simulate a gate was only 172 bytes.

It is interesting to compare performance of SEESIM with that of PROOFS, which used the parallel fault simulation scheme. Table 4 lists the results of applying SEESIM to benchmark circuits by simulating the same number of patterns as that of PROOFS [11]. The simulation times for PROOFS were on a SUN3/280 machine.

Table 3: Run results of SEESIM and the commercial concurrent fault simulator (CFS) on 24 benchmark circuits

Circuit	Fault coverage (%)	CFS		SEESIM		Ratio CFS/SEESIM	
		Time (s)	Memory (Mbyte)	Time (s)	Memory (Kbyte)	Time	Memory
s208	44.1	4	0.35	0.32	17.1	12.6	21.1
s298	48.9	6	0.36	0.67	19.8	9.0	18.5
s344	91.9	7	0.40	0.35	25.5	20.0	16.1
s349	91.5	7	0.40	0.35	25.8	20.0	15.9
s382	20.6	6	0.40	2.03	25.8	2.95	15.8
s386	39.7	4	0.40	0.60	25.8	6.67	15.7
s400	20.3	7	0.40	2.18	26.8	3.21	15.3
s420	40.1	7	0.43	0.73	33.5	9.55	13.1
s444	16.0	7	0.40	2.57	29.6	2.73	14.0
s526	9.7	8	0.43	3.00	32.9	2.67	13.3
s526n	9.7	8	0.43	3.03	32.9	2.64	13.3
s641	76.5	10	0.55	0.58	55.6	17.1	10.1
s713	74.0	11	0.56	0.73	59.8	15.0	9.5
s820	38.2	9	0.52	1.58	52.1	5.68	10.2
s832	37.4	9	0.52	1.62	52.5	5.57	10.1
s838	28.3	17	0.63	1.75	65.8	9.71	9.8
s953	21.2	46	0.73	21.53	65.9	2.14	11.4
s1196	60.3	14	0.65	2.55	81.2	5.49	8.2
s1238	55.7	15	0.64	2.90	82.5	5.17	8.0
s1423	22.9	26	0.80	9.37	102.0	2.78	8.0
s1488	50.2	19	0.83	2.60	96.8	7.31	8.8
s1494	49.2	20	0.83	2.87	96.8	6.98	8.7
s5378	54.1	71	2.44	21.58	363.0	3.29	6.9
s35932	63.1	n.a.	n.a.	189.1	2394.0	n.a.	n.a.
Mean						7.75	12.2

Table 4: Run results of SEESIM and PROOFS

Circuit	PROOFS		SEESIM		Ratio PROOFS/SEESIM	
	Time (s)	Memory (Kbyte)	Time (s)	Memory (Kbyte)	Time	Memory
s208	1.0	80	1.1	16.6	0.91	4.8
s298	1.8	96	2.0	20.3	0.90	4.7
s344	1.3	104	1.2	24.9	1.08	4.2
s382	32.0	112	53.0	33.1	0.60	3.4
s400	24.8	112	31.2	30.7	0.79	3.7
s420	4.1	120	3.6	33.8	1.14	3.5
s444	37.2	120	58.5	35.3	0.64	3.4
s641	2.5	208	1.2	53.3	2.08	3.9
s713	2.6	216	1.4	56.6	1.86	3.8
s820	9.6	192	7.6	56.8	1.26	3.4
s832	9.1	176	7.1	56.6	1.28	3.1
s953	3.5	176	2.5	65.0	1.40	2.7
s1196	6.7	216	3.5	84.9	1.91	2.5
s1238	8.2	216	3.9	85.6	2.10	2.5
s1423	7.0	344	5.7	103.0	1.23	3.4
s1488	23.1	272	24.8	100.0	0.93	2.7
s1494	20.2	272	24.5	99.6	0.82	2.7
s5378	99.5	752	63.6	380.0	1.56	2.0
s35932	340.9	5872	181.3	2472.0	1.88	2.4
Mean					1.28	3.3

The simulated patterns were ATPG-generated. The ratio of simulation times and memory usage for PROOFS to SEESIM are listed in the last two columns of the Table. SEESIM was about 1.3 times faster than PROOFS, and needed about one-third of the memory required by PROOFS. Furthermore, SEESIM can be easily extended to multivalued or higher level simulation because no coding scheme is used, whereas a coding scheme must be used in PROOFS because of its parallel fault treatment.

4 Conclusion

This paper has proposed an SEE concept for sequential circuit fault simulation. The concept is basically a method of dynamic identification of equivalent faults detected by the simulated pattern. It combines the advantages of critical path tracing, the fanout-free region and the dominator concept. These techniques were used only for combinational circuits, but are now applicable to sequential fault simulation with the SEE technique. It is very simple in concept and can therefore easily incorporate other techniques, such as the early stop, fault dropping and the simulation index, to further enhance the simulation speed, and it can be easily extended to the higher level or the multivalued fault simulation. The implemented program, SEESIM, based on the SEE concept, has achieved a simulation speed comparable to the parallel-pattern type of fault simulator. Compared with a state-of-the-art commercial concurrent fault simulator, simulation speed was twice that of the 27 ISCAS benchmark circuits, but it used far less memory. Compared with PROOFS, it also exhibited a better performance. SEESIM has also demonstrated that it uses nearly minimal memory: it uses only one-tenth of the memory of a state-of-the-art commercial concurrent fault simulator.

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 for simulating faults in logic circuits', *IEEE Trans.*, 1972, **C-21**, pp. 464–471
3 ULRICH, E.G., and BAKER, T.: 'The concurrent simulation of nearly identical digital networks'. Proc. 10th Design Automation Conf., 1973, **6**, pp. 145–150
4 FUNATSU, S., WAKATSUKI, N., and ARIMA, T.: 'Test generation systems in Japan'. Proc. 12th Design Automation Conf., June 1975, pp. 114–122
5 EICHELBERGER, E.B., and WILLIAMS, T.W.: 'A logic design structure for LSI testability'. Proc. 14th Design Automation Conf., June 1977, pp. 462–468
6 STEWART, J.H.: 'Application of scan/set for error detection and diagnostics'. Proc. International Test Conf., 1978, pp. 152–158
7 ANDO, H.: 'Testing VLSI with random access scan'. Proc. COMPCOM S'80, 1980, pp. 50–52
8 WAICUKAUSKI, J.A., EICHELBERGER, E.B., FORLENZA, D.O., LINDBLOOM, E., and MCCARTHY, T.: 'Fault simulation for structured VLSI'. Proc. 14th Design Automation Conf., June 1977, pp. 492–494
9 MAAMARI, F., and RAJSKI, J.: 'A fault simulation method based on stem regions', *IEEE Trans.*, 1990, **CAD-9**, (2), pp. 212–220
10 UNDERWOOD, B., and FERGUSON, J.: 'The parallel-test-detect fault simulation algorithm'. Proc. International Test Conf., 1989, pp. 712–717
11 NIERMANN, T.M., CHENG, W.T., and PATEL, J.H.: 'PROOFS: a fast, memory efficient sequential circuit fault simulator'. 27th ACM/IEEE Design Automation Conference, 1990, pp. 535–540
12 CHENG, W.T., and YU, M.L.: 'Differential fault simulation — a fast method using minimal memory'. 26th ACM/IEEE Design Automation Conference, 1989, pp. 424–428
13 HONG, S.J.: 'Fault simulation strategy for combinational logic networks'. 8th Int'l Fault-Tolerant Computing Symposium, June 1978, pp. 96–99
14 ABRAMOVICI, M.A., 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
15 BRGLEZ, F., and FUJIWARA, H.: 'A neutral netlist of 10 benchmark circuits and a target translator in fortran', special session on ATPG and fault simulator, Proc. 1985 Int'l Symp. on Circuits and Systems, Kyoto, Japan, 5–7 June, 1985.
16 BRGLEZ, F., BRYAN, D., and KOZMINSKI, K.: 'Combinational profiles of sequential benchmark circuits'. International Symposium on Circuits and Systems, ISCAS'89, 1989, pp. 1929–1934
17 SCHNURMANN, H.D., LINDBLOOM, E., and CARPENTER, R.G.: 'The weighted random test pattern generator', *IEEE Trans.*, 1975, **C-24**, (7), pp. 695–700
18 ABRAMOVICI, M., KULIKOWSKI, J.J., MENON, P.R., and MILLER, D.T.: 'SMART and FAST: test generation for VLSI scan-design circuits', *IEEE Des. Test. Comput.*, 1986, **3**, (4), pp. 43–54
19 SCHULZ, M.H., TRISCHLER, E., and SARFERT, T.M.: 'SOCRATES: a highly efficient automatic test pattern generation system', *IEEE Trans.*, 1988, **CAD-7**, (1), pp. 126–137
20 CHENG, W.T., and CHAKRABORTY, T.J.: 'GENTEST: an automatic test-generation system for sequential circuits', *IEEE Comput.*, 1989, **22**, (4), pp. 43–49
21 AGRAWAL, V.D., CHENG, K.T., and AGRAWAL, P.: 'A directed search method for test generation using a concurrent simulator', *IEEE Trans.*, 1989, **CAD-8**, (2), pp. 131–138