Fast Fault Simulation for BIST Applications

Chen-Pin Kung, Chun-Jieh Huang & Chen-Shang Lin
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC

Abstract

Fault simulation is essential to design a high fault-coverage BIST. The simulation is characterized by combinational fault simulation and signature computation with a large amount of test patterns. In this paper, a fast fault simulator BISTSIM for BIST is developed. For the combinational fault simulation, a novel demand-driven logic simulation algorithm is proposed. Moreover, efficient fault propagation methods are incorporated into BISTSIM. The experimental results show that the proposed fault simulator delivers better performance than FSIM[8] about 2 to 3 times for circuits with a large number of test patterns. For signature evaluation of MISR to determine the aliasing, two efficient simulation methods, bit-array computation and parallel-pattern sequential simulation, are proposed. The resultant BISTSIM outperforms the fast fault simulator HOPE1.1[12] with an average speedup ratio of 10.

1 Introduction

Built-in self-tests are becoming popular for VLSI testing. The typical configuration employs linear feedback shift registers (LFSRs) to generate pseudorandom pattern for the combination circuits and to compact the resultant responses. It features the low hardware cost and high fault coverage in general. However, some circuits contain random-pattern resistant faults which are difficult to be detected in a reasonable number of patterns. To detect these faults, many strategies have been developed, which include multiple polynomial, multiple seed [16], weighted random testing[17, 18], and adding deterministic test patterns[15]. All the above methods demand an efficient fault simulator to evaluate the fault coverage of an enormous amount of test patterns. Moreover, one may need to run many fault simulations in order to choose a good test strategy. Hence, an efficient fault simulator may speed up the design process.

The efficiency of fault simulation in a BIST design environment is determined by the combinational fault simulation and the signature computation for aliasing. Previous studies on these two areas will be briefly reviewed in the following.

Parallel pattern single fault propagation (PPSFP)[1] is widely used in combinational circuit fault simulation. Many proposed fault simulators incorporate PPSFP with other sophisticated techniques such as test detect[4], critical path tracing[2, 3], stem region[5-7] and dominator concept[3-8]. These techniques reduce the simulation time by utilizing informations of previous simulated faults. Thus the number of gate evaluations for faulty circuit simulation is reduced. The fault simulators Tulip1[6], and Tulip2[7] further reduce the logic simulation time by dynamically storing necessary gates for logic simulation. Lee and Ha[8] observed that there is still room for improvement because these algorithms have the drawbacks of repeatedly simulation of stem region and/or inefficient for circuits with high depth. They proposed a new fault simulator FSIM which incorporate new heuristics with PPSFP. The major features of FSIM include simulation in forward levelized order and desensitizing unnecessary test patterns. They reported very good performance by FSIM. These algorithms, although efficient, are generally targeted for fault simulator in a deterministic ATPG environment. However for circuits with BIST, the efficiency of these fault simulators may not be sufficient. Since the test length of the circuit with BIST is quite lengthy in general, most of faults are detected and dropped from simulation in the small beginning part of test patterns, and only a small portion of faults are simulated with the remaining major part of patterns. A typical result is that the similarity between different faults becomes less due to sparser fault sites. Thus the benefit of above sophisticated algorithms to exploit sharing common informations among faults is reduced. A fault simulator targeted for BIST will be more efficient under such circumstance.

Multiple input signature register (MISR) is widely used as the signature analyzer to compact the output responses of the circuit under test (CUT). With MISR, the outputs of CUT are fed to inputs of MISR. During the test mode, the MISR compact the output responses and the final state of the MISR serves as the signature which is then checked to determine whether the CUT passes or fails under the test. Hence, testing with signature analyzer has the merits of simplicity and low hardware cost because that there is no need to store the entire responses of test patterns. However, because of compaction, some faults may produce the same signature as fault free circuit. Such a case is named an *aliasing*. The number of aliasing depends on the size of LFSR and the characteristic polynomial. One may compute the aliasing with the length of MISR or with the signal probability. However if high fault coverage is required, one then has to compute the exact number of aliasing or to determine how many test clocks required to achieve the desired fault coverage. This means the simulation or computation the exact signature of each fault is required. Since MISR is by nature a sequential circuit, the signature together with combinational circuit simulation can theoretically be done with any existing sequential circuit fault simulator[13]. However, as will be shown in our experimental results, this is extremely time consuming. This motivates us to develop a fast fault simulator with separate combinational fault simulation and signature simulator for BIST design environment.

In this paper, we present a fast fault simulator, BISTSIM, targeted for BIST environment. BISTSIM consists of two components: a combinational fault simulator and a signature evaluator. For the combinational fault simulation, logic simulation time is reduced by the novel demand-driven logic simulation. Furthermore, to reduce overhead of net-list traversal, multiple word simulation is incorporated in BISTSIM. BISTSIM also adopts the simulation IDs technique proposed in PROOFS[9] to further improve the simulation speed. The experimental results show that BISTSIM is faster than FSIM by about 2 to 3 times for random-resistant circuits which require a large amount of test patterns.

For the signature evaluation, two algorithms, bit-array computation and parallel-pattern sequential simulation are presented.

0-8186-7129-7/95 $4.00 © 1995 IEEE
The Fourth Asian Test Symposium

The bit-array computation compresses the output responses and state of MISR and the value of feedback state (coefficients) into bit array. The next state of MISR is then evaluated with parallel word operation on these bit arrays. Alternatively, parallel-pattern sequential simulation evaluates signature by treating MISR as a sequential circuit and incorporates parallel pattern simulation with multiple iterations. To reduce the number of iterations, simulation of gates is ordered based on the structure of MISR. The experimental results show that the resultant BISTSIM outperforms the well-known sequential fault simulator HOPE1.1 with an average speedup ratio of 10.

The organization of this paper is as follow. In Section 2, the combinational fault simulation of BISTSIM, BISTSIM_c, will be described. The experimental results of BISTSIM_c are shown in Section 3. In Section 4, two signature evaluation algorithms will be described. The comparison results of these two algorithms and the resultant fault simulation BISTSIM_s are given in Section 4. Finally, the conclusions are given in Section 5.

2 Combinational fault simulation of BISTSIM

In this section, we present a fast parallel pattern fault simulator, BISTSIM_c targeted for BIST design environment. BISTSIM_c is the combinational fault simulation component of BISTSIM.

2.1 Basic simulation strategy of BISTSIM_c

BISTSIM_c is developed based on the behavior of fault simulation for circuits with BIST. In BIST, the pseudorandom pattern generated from LFSR may not detect all of detectable faults in a reasonable amount of test clocks. Even with weighted pseudorandom pattern, the test length is still long to detect some random-pattern resistant faults. A typical scenario of fault simulation is that after most of faults are detected by the first small part of test patterns, the remaining faults are redundant faults or hard-to-detect faults. Hence during the simulation, these faults are either not activated or the propagation zones of their fault effects are small. As a result, most of the simulation time will consumed by the logic simulation. Since the remaining faults are small in number and their propagation zones are small. The benefit of sophisticated algorithms such as stem region, test-detect and dominator concept that exploit sharing common informations between faults is reduced. Hence these techniques are not suitable for circuits with BIST. Based on this observation, we develop the novel demand-driven logic simulation to reduce logic simulation time and an efficient propagation method for a faster simulation.

The basic simulation algorithm of BISTSIM_c is as follows. BISTSIM_c is a combinational circuit fault simulator based on PPSFP in which one fault is simulated with a packet of patterns in parallel. For each packet, first, logic simulation is done in the area of gates driving the faults undetected so far, and the fault-free values of gates behind this area are left to be evaluated during the fault effect propagation in the following steps. First, each fault is simulated to its fanout free region (FFR) output with single fault propagation. If a fault propagates to its FFR output, its fault effect on the stem will be stored. After all faults are simulated to their FFR outputs, it is followed by the surrogate fault propagation procedure. In this step, a surrogate fault is activated to propagate the fault effects in the FFR. BISTSIM_c adopts the concept of desensitization of unnecessary test patterns proposed in FSIM, simulation IDs techniques and increasing packet size for faster simulation. These techniques will be further described in the following subsections. First the demand-driven logic simulation will be described.

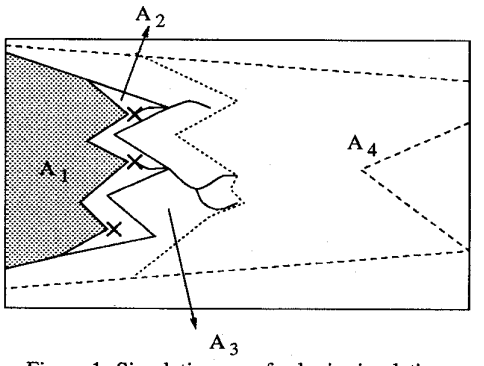

Figure 1. Simulation area for logic simulation

2.2 Demand-driven simulation

The idea of demand-driven logic simulation is to minimize the time for logic simulation by evaluating the fault-free value on a gate only when it is indispensable. The demand-driven simulation avoids unnecessary gate evaluations which may occur in the previous works.

The concept of reducing logic simulation area have been developed in Tulip and improved in Tulip2. These techniques are based on the observation that during fault simulation, the fault-free value on a gate is only required for activating a fault effect and propagating the fault effect. Hence the gates that required to evaluate their fault-free statuses are either driving the fault sites or driving the fault effect propagation paths. The simulation area is shown in the area A_4 indicated with long dashed line in Fig. 1. In Tulip1, these gates are stored for logic simulation. As faults are detected and dropped during the simulation, the algorithm keeps track of the simulation area. When more faults are detected, the simulation area is reduced and hence the logic simulation time is reduced. However, if a fault can not be activated by the currently simulated pattern, then the area of gates driving the fault propagation paths are not necessary. To further reduce the simulation area, in Tulip2, the reduction of logic simulation is achieved in two steps. First, logic simulation is done in the area of gates driving faults undetected so far to the corresponding FFR outputs as shown in area A_2 of Fig. 1. It is followed by the critical path tracing in FFRs in the area. If no such fault propagates to their FFR output, then there is no need to continue the logic simulation. Thus the simulation area of $A_4 - A_2$ in Fig. 1 is further reduced. However, there is still unnecessary gate evaluation of the above method. If a fault propagates the fault effect to its FFR output but the fault effect can only be further propagated in short distance, simulating the whole area behind the fault site is still a waste.

To minimize the number of gate evaluations for logic simulation, a more efficient demand-driven algorithm is developed in BISTSIM_c. The logic simulation in BISTSIM_c is performed as follows. First, the area of gates driving the undetected faults is simulated. This area is shown as A_1 in Fig. 1. The size of the area is smaller than the simulation area A_2 in the first step in Tulip2, because the area in Tulip2 contains the whole FFR containing the fault site, while the area in BISTSIM_c contains only gates driving the fault site. Gates in this area are maintained in a queue and evaluated in forward levelized order. After simulating the packet of patterns, the queue containing gates for logic simulation is updated if any newly detected fault is identified. For the area of gates outside A_1, the activation of logic simulation is determined in faulty circuit simulation. During the faulty circuit simulation, if the fault-free value of a gate is required, then a backward traversal procedure is invoked to evaluate all the gates

required the current fault-free values. The simulation algorithm is named as demand-driven since all the fault-free values are evaluated only in demand during this stage. Since the fault-free values of gates driving the fault effect propagation paths are only evaluated during fault effect propagation. If the propagation path of a fault is short, the number of gates for logic simulation is reduced. In Fig. 1, if the propagation paths of the fault effects are shown as the curved lines. Then only the area A_3 is required for logic simulation. Thus the waste simulation for the area A_4 - A_3 in Tulip2 is avoid, and the logic simulation area will be smaller than Tulip2. Since the fault effects of random pattern resistant faults can only be propagated in short paths in general, the benefit of demand-driven simulation becomes more evident after most of faults are detected. This indicates that demand-driven logic simulation is very efficient for circuits with BIST in which the test length is long and most of run time is consumed by the logic simulation.

In demand-driven simulation, it needs to check if a fault-free value on a gate is correctly representing the current status. To solve this problem, we extend the concept of simulation ID proposed in PROOFS[9] to logic simulation. In BISTSIM_c, a simulation ID record is associated in each gate and a global simulation ID represents the correct pattern stamp. The global simulation ID is incremented at each simulation of a packet patterns. The value of each gate's simulation ID is set to the value of global ID after evaluation. Thus, during the faulty circuit simulation, the check of whether the logic value of a gate is correct is done by comparing the value of global simulation ID with the ID record associated with the gate. If the values of simulation IDs are different, the demand-driven simulation is invoked to simulate all the required gates. The overhead of our demand-driven simulation is that before simulating a gate, the values of simulation ID should be compared. However, the overhead is generally negligible.

2.3 Fault effect propagation and multiple word simulation

BISTSIM_c employs FFR processing to reduce the number of gate evaluations during faulty circuit simulation. At first, each fault is propagated to its FFR output with PPSFP. If its fault effect is propagated to primary outputs then the fault is detected and dropped from further simulation. On the other hand, if the fault effect is propagated to a stem, then the propagated fault effect are stored. The propagated fault effects on FFR outputs will be further propagated in the next step, the surrogate fault simulation. After all faults have been simulated, the fault effects that propagated to stems are now simulated. In surrogate fault simulation, a stem fault is injected to propagate the fault effects in its FFR. BISTSIM_c adopts the concept of desensitizing unnecessary test patterns proposed in FSIM. That is, unlike the injection method proposed in Parallel-Test-Detect[4] which injects fault effects by complementing all bits of the fault-free value. In BISTSIM_c, the injected fault effects are the union of the propagated fault effects. Thus only the sensitized bits are complemented as in F-SIM. Therefore unnecessary propagations due to complementing all bits can be avoided. Injecting only sensitive bits on a stem not only reduces gate evaluations for fault propagation but also for the logic simulation. Since a logic value is evaluated only on demand, pruning unnecessary propagation path will certainly reduce the load for logic simulation.

In BISTSIM_c, all the fault effect propagation inside an FFR and surrogate fault simulation are performed with single fault propagation. The techniques of critical path tracing and dominator concept are not employed. The reason is that the benefit of these algorithms is reduced when faults to be simulated is small in number. Furthermore single fault propagation is able to minimize the area for logic simulation and it is suitable for multiple word simulation. The overhead of unnecessary gate evaluations within a FFR is eliminated by the multiple word simulation and reduced gate evaluations for logic simulation. The efficiency of single fault propagation is further improved with the simulation IDs technique. BISTSIM incorporates the simulation IDs technique not only in demand-driven logic simulation but also on the fault effect propagation. With this technique, the faulty status and fault-free value of a gate are stored in the different computer words. The simulation ID associated with a gate is checked during fault effect propagation to see if the faulty status word represents the current fault effect. Hence, the fault propagation can be proceeded more efficiently because there is no need to restore the fault-free status after simulating a fault.

Another feature of BISTSIM is that it incorporates the technique of increasing packet size for faster performance. In parallel pattern simulation a packet of patterns is simulated in parallel. As the packet size increases, the performance is generally improved owing to increased parallelism. The recently developed simulators use the packet size of 32 as the word length of most workstations is 32. The packet size can be increased by using multiple words as a packet. Increasing packet size may increase the parallelism and reduce the time for net-list traversal. However it also increases the processing time of each gate evaluation. Hence, for easy-to-detect faults, increasing packet size may introduce more overhead than single word simulation. Increasing the packet also increases the memory requirement which may cause many page faults during simulation and decrease the performance. Nevertheless, it may not be a serious problem for modern workstations with large memory as long as the packet size is not excessive.

In FSIM, various packet sizes were experimented. However, the results of FSIM shows that multiple word simulation is not beneficial. This is because FSIM incorporates sophisticated techniques such as critical path tracing and dominator concept with PPSFP. Increasing packet size with multiple words complicates these techniques and thus the code overhead on multiple word simulation exceeds the benefit of parallelism. Furthermore, the benefit of these techniques is reduced when the number of undetected faults is small and increasing the packet size can not have a major improvement in FSIM. In BISTSIM_c, the critical path tracing for processing faults inside an FFR and the dominator concept are not employed. Since the fault propagation in BISTSIM_c is compatible to PPSFP, the code overhead will not exceed the benefit of parallelism and the simplicity of fault propagation makes the increasing packet size become more efficient. As will be shown in the experimental results, increasing the packet size may obtain a speedup of more than 2 times.

Since increasing packet size may incur overhead for easy-to-detect faults which becomes apparent when the number of test patterns is small. Hence BISTSIM_c employs two-phase simulation to reduce this overhead. In the first phase, multiple word simulation is only performed on logic simulation and fault simulation uses the packet size of one word. When the number of faults detected by the current packet is reduced to a pre-determined number, 5% of the number of faults in our implementation. The simulation strategy is switched to phase 2. In phase 2, both logic simulation and fault simulation use multiple-word packet. Thus with two-phase simulation, both easy-to-detect and hard-to-detect faults can be simulated efficiently. BISTSIM_c employs the simulation IDs technique proposed in PROOFS and the packets used for logic simulation and faulty circuit simulation are represented in different words. In our implementation, the computer words representing packets of fault-free value and faulty value are allocated in a continuous region. The phase switching can be easily done by adjusting the pointer of the packet. For example, if 32 words are allocated, the packet sizes are arranged as 31 words for fault-free circuit and 1 word for faulty circuit in phase 1. In phase 2, both the packets for fault-free circuit and faulty circuit have the size of 16 words.

Table 1. CPU time for various packet size

Circuit	#vec	Packet size (#words)						
		1	2	4	8	16	24	32
c2670	3200	1.78	1.75	1.37	1.27	1.50	1.80	2.12
c2670	32000	14.95	13.97	9.05	7.35	6.68	6.75	6.98
c2670	320000	137.47	127.97	81.53	64.38	55.72	53.85	53.08
c7552	3200	3.65	3.40	2.80	2.73	2.98	3.33	3.75
c7552	32000	21.67	18.52	12.73	10.73	9.88	9.90	10.32
c7552	320000	174.30	144.67	95.33	76.32	66.60	64.17	63.95
c_s9234	3200	10.50	9.43	8.30	8.98	9.57	9.87	11.48
c_s9234	32000	62.10	54.85	37.08	30.98	28.12	27.23	28.50
c_s9234	320000	421.22	353.52	224.82	172.52	146.63	137.75	137.82
c_s15850	3200	9.93	8.80	7.82	7.88	10.13	11.08	10.12
c_s15850	32000	54.03	43.43	29.98	24.43	23.70	24.22	24.62
c_s15850	320000	417.18	319.82	206.05	154.12	131.52	132.80	121.18
c_s38417	3200	21.05	20.03	16.52	18.57	18.93	22.03	25.18
c_s38417	32000	109.43	95.67	67.02	58.80	53.85	55.92	58.75
c_s38417	320000	582.58	501.25	345.65	280.70	245.30	242.67	247.65
c_s38584	3200	14.80	16.13	13.40	13.33	14.37	16.52	18.40
c_s38584	32000	71.25	77.67	54.35	44.23	42.32	44.27	46.57
c_s38584	320000	560.68	601.68	401.78	300.37	257.32	250.07	243.13

Table 2. Performance of FSIM and BISTSIM

Circuit	#vec.	Fault cov.	CPU time			Speedup ratio
			FSIM	BISTSIM_1	BISTSIM_c	
c2670	3200	84.75	1.62	1.78	1.50	1.08
c2670	32000	85.04	12.97	14.95	6.68	1.94
c2670	320000	87.08	114.32	137.47	55.72	2.05
c7552	3200	93.44	4.13	3.65	2.98	1.39
c7552	32000	94.97	23.75	21.67	9.88	2.40
c7552	320000	96.76	205.28	174.30	66.60	3.08
c_s9234	3200	79.50	10.43	10.50	9.57	1.09
c_s9234	32000	86.79	55.68	62.10	28.12	1.98
c_s9234	320000	91.32	359.55	421.22	146.53	2.45
c_s15850	3200	89.77	13.95	9.93	10.13	1.38
c_s15850	32000	92.58	62.58	54.03	23.70	2.64
c_s15850	320000	94.86	450.42	417.18	131.52	3.42
c_s38417	3200	89.32	27.85	21.05	18.93	1.47
c_s38417	32000	94.79	142.75	109.43	53.85	2.65
c_s38417	320000	98.31	737.67	582.58	245.30	3.01
c_s38584	3200	92.40	24.88	14.80	14.37	1.56
c_s38584	32000	95.04	99.77	71.25	42.32	2.36
c_s38584	320000	95.69	618.62	560.68	257.32	2.40

In the next section, the experimental results of BISTSIM_c will be reported.

3 Experimental Results of BISTSIM_c

BISTSIM_c is implemented with C and run on SUN Sparc station 2 with 32M bytes memory. Its performance is evaluated with those circuits containing random-pattern resistant faults from ISCAS'85 benchmark[20] and full-scanned ISCAS'89 sequential benchmark[21]. The performance of BISTSIM_c will be compared with FSIM which delivers best performance to our knowledge. Each fault simulator is run with pseudorandom patterns of lengths 3200, 32000 and 320000 to evaluate the performance in detail.

To examine the effect of multiple word simulation, the experiment of various packet sizes is conducted first. The results are shown in Table 1. In this table, the packet size is reported by the number of computer words. As the packets used for logic simulation and faulty circuit simulation are represented in different words. The total number of computer words used as packets for both logic simulation and faulty circuit simulation is two times of the reported value. From table 1, it can be seen that the performance is improved with increasing packet size generally. However, the degree of improvement depends on the size of packet and the test length. From the table, it can also be seen that the improvement with multiple words is more evident for long test length. For 320000 patterns, the performance can be speeded up about 2 to 3 times. The results show that in general a significant improvement is obtained with packet sizes in the range of 4 words to 24 words. And the performance improvement is saturated at the packet size of about 16 words. When the packet size is increased to 32 words, the improvement begins to degrade due to page faults. The performance improvement of increasing packet size therefore depends on the size of memory that a workstation contains. However, it is clearly from the table that performance can be improved with increasing packet size even for the large circuits.

In Table 2, the performance of BISTSIM is compared with the fault simulator, FSIM. The results of BISTSIM with packet size of 16 words (BISTSIM_c) and the results of BISTSIM with packet size of single word (BISTSIM_1) are taken from Table 1 for comparison. The speedup ratios of BISTSIM_c over FSIM are given in the last column. From Table 2, it can be seen that BISTSIM_1 delivers better performance than FSIM except for circuits c2670 and c_s9234. The results illustrate the efficiency of demand-driven logic simulation, since FSIM employs the algorithm proposed in Tulip1 to reduce the logic simulation time while BISTSIM_1 employs demand-driven logic simulation to reduce the load for logic simulation. For circuits c2670 and c_s9234, the performance of BISTSIM_1 is slower than FSIM because BISTSIM_1 does not incorporate sophisticated techniques such as critical path tracing and dominator concept as in FSIM. For circuits with high depth, these techniques deliver better performance. However they become less desirable when multiple words are employed as in BISTSIM_c. Comparing FSIM with BISTSIM_c which incorporates multiple word simulation, it is clearly that BISTSIM_c outperforms FSIM by about 2 to 3 times for long test length.

In summary, BISTSIM_c is suitable for BIST circuits for combinational fault simulation since it delivers a significantly better performance with more than thousands test patterns. Its efficiency is owing to novel demand-driven logic simulation, simulation IDs technique and the multiple word simulation. Furthermore, the result indicates that although some sophisticated algorithms such as critical path tracing and dominator concept are not exploited in BISTSIM_c, the performance can be speeded up with increasing packet size because BISTSIM_c preserves the simplicity and flexibility of PPSFP.

4 Signature evaluation

In this paper we consider the signature analyzer that implemented with multiple input signature register (MISR) because it is widely used. Without lost of generality, the type-2 MISR is considered. The structure of a MISR is shown in Fig. 2. In Fig. 2, the value of Q_n is fed to a flip-flop Q_i depending on if the value of the coefficient C_i is 1. For simplicity, we named the flip-flop that feeds its value to the other preceding flip-flops as the *feedback flip-flop* (Q_n in Fig. 2). The value of a flip-flop is evaluated with exclusive-or operations on values of primary output, preceding flip-flop and the feedback flip-flop.

The simulation of signature is known as to be time consuming because of following reasons. First, to simulate the signature of each fault, the output response of a faulty circuit at each clock has to be determine. This means that no fault can be dropped during simulation. Furthermore, it has to retrieve the values at circuit outputs and present state of MISR to compute the next state

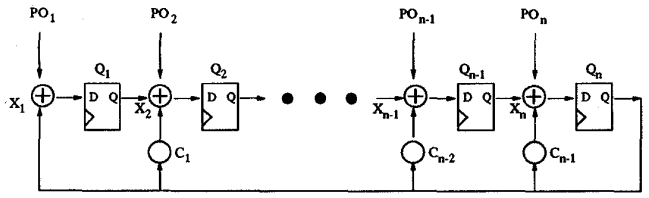

Figure 2. The structure of a type-2 MISR

of MISR. A large number of gate evaluations is required if the length of MISR is long. Second, if a fault is not activated by the current pattern but the state of MISR is different from the fault-free value due to fault effects in previous time frames, the state value at MISR still must be computed. In our experience, the time for signature evaluation is significantly longer than the time for combinational fault simulation alone, which may take more than 90% of the entire simulation time.

As the structure of a MISR is by nature sequential, the signature evaluation can be done with any existing sequential circuit fault simulator. With this approach, the combinational CUT and the MISR is treated as a sequential circuit, and the signature evaluation is implicitly done by simulating the state response of this circuit. However, this approach is not efficient as will be illustrated in the results reported in the next section. Three factors make the sequential circuit simulation inefficient. First, combinational fault simulation in the CUT with sequential fault simulator is not efficient, since many sophisticated techniques such as FFR processing, desensitizing unnecessary path dedicated for combinational circuits are not suitable for a sequential circuit. Second, sequential circuit fault simulator uses 3-valued logic to handle unknown initial state. However, the initial state of a signature analyzer is always set to a deterministic state and hence, faster 2-valued logic evaluation is enough for signature evaluation. Finally, as the signature is implicitly simulated with state responses. The regularity of the structure of a MISR is not considered, which can be exploited to our advantage. Therefore, simulating signature with existing sequential circuit fault simulators is not a good choice.

In this paper, we simulate the CUT with combinational fault simulator as described in BISTSIM_c without fault dropping and present two efficient methods for signature evaluation. One is the bit-array computation and the other, the parallel-pattern sequential simulation. Bit-array computation simulates the signature of a circuit by first compressing the output response and the state of MISR into bit arrays, and then evaluating these bit arrays with parallel word operation. On the other hand, parallel-pattern sequential simulation simulates signature by treating the MISR as a sequential circuit and simulating with parallel pattern simulation of multiple iterations. To reduce the number of iterations, gates of MISR are simulated in a dedicated order. The experimental result shows that both these two methods deliver better performance than existing sequential circuit fault simulators. These two signature evaluation methods will be described in the following subsections. Our resultant fault simulator with signature evaluation will be called BISTSIM_s.

4.1 Bit-array computation

The bit-array signature computation is quite straightforward. It computes the state of the MISR from the output response, the present state of MISR and the coefficients of MISR. For efficient performance, these values are arranged into bit-array and evaluated with parallel word operation. Thus the number of gate evaluations of signature simulation is reduced to the number of primary outputs divided by the word length of the workstation. The simulation procedure is accomplished in following steps. First, the output response of a circuit and its present state in MISR at a clock is compressed into bit arrays. Then the next state of MISR can be evaluated with following equation.

$$NEXT = PO \oplus STATE \oplus CO, \qquad (1)$$

where NEXT, PO, STATE and CO represents the bit arrays for next state, output response, present state and coefficient of the MISR respectively and $\oplus$ represents the exclusive-or operation. The values of the bit-array CO can only be zero or one depending on the coefficients of polynomial.

The advantage of bit-array computation is that its performance is independent of the realization of a MISR. However, the computation time is proportional to the test length and the length of MISR which in general, is equal to the number of primary outputs. Clearly, it is less efficient for a circuit that contains a large number of primary outputs. Hence we develop the parallel-pattern sequential simulation method to evaluate signature.

4.2 Parallel-pattern sequential simulation

The parallel-pattern sequential simulation method evaluates the signature by treating the MISR as a sequential circuit with the primary outputs of CUT as the primary inputs of the MISR. Then the signature evaluation is performed in parallel-pattern sequential circuit fault simulation which is based on PARIS[10]. It simulates a packet of patterns for each circuit, fault-free and faulty with following steps. First, the present state of a circuit is loaded in the first bit of the packet. Then the MISR is simulated with parallel pattern simulation. For a flip-flop, the output value is determined by shifting the input value one bit due to one-time-frame delay. The above simulation step is repeated for each packet until there is no change on the state of flip-flops, or the state converges. The resultant value is then the correct value. The number of iterations to convergence determines the efficiency of parallel-pattern simulation.

In parallel-pattern simulation, since at the beginning of a simulation step, all the bits except the first on the output of a flip-flop are unknown. The number of iterations can be reduced if appropriate values are set for these bits. In PARIS, the values on these bits are set with the strategy called *heuristic look-ahead of signal values*. For logic simulation, the previous states are retained at a new simulation step, and for faulty circuit simulation, these states are set to either the fault-free values at this step or the faulty values at the last step depending on the degree of difference between the faulty values and the good values at the end of the last step. This state guessing method is efficient for general circuits that contain random logic because the state difference between the faulty circuit and good circuit or the state difference of a faulty circuit between two time frame are small in general. However, since the purpose of MISR is to compress input sequences into signature, the similarity of states between different circuits or time frames is scarce. As a result, the state guessing method is not suitable for signature evaluation. It may require a large number of iterations to obtain the correct result.

As the state guessing approach is not suitable for MISR. In our simulator, the regularity of the structure of a MISR is taken into consideration to reduce the iteration number. By inspecting the structure of MISR, it can be seen that the value of a flip-flop is either shifted from the previous flip-flop or the result of the previous flip-flop exclusive-or with the feedback flip-flop. Simulating gates in appropriate order will speed up the performance. For example, the original simulation order in PARIS is $\{X_1, X_2, \cdots X_n\} < \{Q_1, Q_2, \cdots Q_n\}$ for the MISR in Fig. 2. The symbol $<$ represents the simulation order and the elements

in the braces can be simulated in parallel. In the worst case, each iteration can only obtain one bit of the correct value. Under such circumstance, the parallel pattern simulation will be less efficient than single pattern simulation due to overhead of multiple iterations. In our simulator, the number of iterations is reduced by simulating gates of MISR in a dedicated order. The simulation order in our simulation is $X_1 < Q_1 < X_2 < Q_2 < \cdots X_n < Q_n$. The simulation order is similar to the concept of strongly-connected-circuit (SCC) proposed in COMBINED[11]. However, the proposed algorithm in this paper is dedicated for signature evaluation. With such gate ordering for simulation in MISR, the number of iterations is less than that in PARIS. Furthermore, we observe a property that the number of iterations depends on the structure of the MISR, ie, the polynomial used and its realization. The property is as follows.

Property 1: Let Q_n be the feedback flip-flop, W be the packet size and k be the smallest number of flip-flops between Q_n and the stages with non-zero feedback coefficient. Then the maximal number of iterations during parallel pattern simulation is $\lceil \frac{W}{k} \rceil + 1$.

The property can be illustrated with the state response of Q_n in Fig. 2 because that the state response of a flip-flop in MISR depends on the value of feedback flip-flop Q_n. Let the state value of Q_n at time t is $Q_n(t)$. Before simulating MISR, the initial state or the final state of previous packet of patterns has to be loaded. Thus the value of each flip-flop at time t is correct. By inspecting the structure of MISR, it can be seen that the state values $Q_n(t+1), Q_n(t+2), \cdots Q_n(t+k-1)$ are obtained by shifting the state values of preceding flip-flops at time t, ie, the values of $Q_{n-1}(t)$, $Q_{n-2}(t) \cdots Q_{n-k+1}(t)$. And $Q_n(t+k)$ is determined by the values $Q_n(t)$ and $Q_{n-k}(t)$. Hence if gates in MISR are simulated with the proposed order, at the end of the first iteration, the first k bits in the packet of state Q_n are correct. At the second iteration, the correct state values $Q_n(t+k+1), Q_n(t+k+2), \cdots Q_n(t+2k)$ will be obtained because these values have been determined from the correct values at time $t+1, t+2, \cdots t+k$. From the observation, it can be seen that each iteration obtains correct k bits of a packet. Hence, the maximal number of iterations to obtain all the correct values of Q_n is $\lceil \frac{W}{k} \rceil$. Since it requires another iteration to obtain values of all remaining flip-flops in MISR, the maximal number of iterations for simulating the MISR is $\lceil \frac{W}{k} \rceil + 1$.

The property reveals that the performance of sequential simulation method can be predicted in a priori based on the polynomial and realization of the MISR. One interesting observation can be obtained from the property. A primitive polynomial of the MISR can have two realizations based on different labeling orders since a primitive polynomial has a corresponding reciprocal polynomial. For example, the feedback flip-flop of a MISR with the polynomial of $x^7 + x + 1$ can be either Q_7 by labeling from left to right or Q_1 with the reverse labeling order. The compressing ability of these two realization are equal. However, the the proposed method delivers quite different performance for these two realizations. The value k of the first realization is 6 and 1 for the other. Hence according to the property, one needs at most 7 iterations to get the correct results for the first case and 32 iterations for the other.

5 Experimental Results of BISTSIM_s

In this section, the performance of proposed two methods for signature evaluation will be evaluated. Due to the unavailability of high-order primitive polynomials, the evaluation is concentrated on ISCAS'85 benchmark circuits. For each circuit, a type-2 MISR with length equal to the number of primary outputs is used as the signature analyzer. A primitive characteristic polynomial is chosen from Reference [19] based on the length of MISR. Throughout all the experiments, the initial state of MISR is set to 0, and its realization is according to the coefficient ordering from left to right.

The experimental results are shown in Table 3. In this table, the polynomial of a MISR is given in the second column. The simulation times for combinational fault simulation without fault dropping and two proposed signature evaluation algorithms are listed in columns 5-7. To evaluate the efficiency of proposed algorithm, two sequential circuit fault simulators, HOPE1.1[12] and PARIS* are also experimented. HOPE1.1 is a simulator exploiting parallel fault simulation, and PARIS* is an our implemented simulator according to the description of [10] that exploits parallel pattern simulation. From the results, it can be seen that both bit-array computation and parallel-pattern sequential simulation proposed in this paper deliver significantly better performance than HOPE1.1 and PARIS*. The average speedup ratios of proposed methods over HOPE1.1 are 4.29 and 10.77. The performance of PARIS* is slower than HOPE1.1 because it is hard to get a good state guessing and thus requires a large number of iterations for a packet simulation. The result clearly indicates that evaluating signature with an existing sequential circuit fault simulator is not a good choice.

Comparing the performance of proposed two simulation methods, it can be seen that the parallel-pattern sequential simulation is much more efficient than the bit-array computation for the given MISR realization. The performance of the parallel-pattern sequential simulation depends on the polynomial as discussed in the above section. For circuits c880, c1908, c2670 and c7552, the parallel-pattern sequential simulation method outperforms bit-array simulation about 4 to 5 times. By inspecting the polynomial, it can be seen that the maximal number of iterations is 2 or 3 for each packet simulation. Hence, the parallel-pattern sequential simulation achieves high performance. For circuits c499, c1355 and c6288 the speedup ratios are reduced because it requires about 9 iterations to get the correct results for each packet of simulation. Although the performance of parallel-pattern sequential simulation method depends on the polynomial and realization of a MISR, its performance can be predicted based on Property 1. Hence, one can choose an signature evaluation method according to the structure of MISR.

The results in Table 3 indicate that the two proposed signature evaluation algorithms is more efficient than existing sequential circuit simulators. However, the results also show that the simulation time for signature evaluation is far more than the time for combinational fault simulation even without fault dropping. For circuits such as c2670, c5315 and c7552, the time for signature evaluation takes more than 90% of the entire simulation time even with bit-array computation. Therefore it clearly deserves our development of efficient algorithms.

6 Conclusions

BIST is widely used in VLSI testing owing to its simplicity and low hardware cost. To ensure the test quality of BIST, one needs to run iterative fault simulations with a large amount of test patterns to evaluate the test strategies during the design process. Furthermore, the exact signature of each fault must be calculated to guarantee that the aliasing does not affect test quality. Therefore efficient combinational fault simulation of large test sets and fast signature evaluation are essential to speed up the BIST design process. In this paper, we have developed an efficient fault simulator, BISTSIM, for such requirements.

BISTSIM consists of two components: a combinational fault simulator and a signature evaluator. For the combinational fault

Table 3. CPU times with 3200 test patterns

Circuit	Polynomial	PARIS*	HOPE1.1	Proposed algorithms			Speedup over HOPE1.1	
				Fault sim.	Bit-array	Parallel pattern	Bit-array	Parallel pattern
c432	7: 1 0	148.47	84.82	5.32	16.42	6.22	3.90	7.35
c499	32: 28 27 1 0	774.00	347.38	10.25	78.9	44.08	3.90	6.39
c880	26: 8 7 1 0	892.62	381.23	9.07	80.57	21.23	4.25	12.58
c1355	32: 28 27 1 0	1746.4	772.97	19.68	163.0	92.8	4.23	6.87
c1908	25: 3 0	1218.8	887.80	32.25	158.1	39.97	4.66	12.29
c2670	140: 29 0	7986.3	4186.9	52.53	1025.9	234.8	3.88	14.57
c3540	22: 1 0	2290.7	1513.8	75.31	243.0	69.07	4.76	10.48
c5315	123: 2 0	16296	8781.4	123.9	2044.2	452.3	4.05	15.24
c6288	32: 28 27 1 0	11543	6195.3	368.2	817.4	515.4	5.23	7.01
c7552	108: 31 0	18676	10398	157.6	2406.3	542.0	4.06	14.86
Average speedup ratio							4.29	10.77

simulation, most of previous works on fault simulation exploited sharing common informations between faults. As it needs a large number of test patterns in BIST to detect pseudorandom pattern resistant fault, the benefit of such techniques becomes significantly degraded and logic simulation time becomes more pronounced due to sparse faults. In BISTSIM, logic simulation time has been reduced by the novel demand-driven logic simulation. Furthermore, to reduce overhead of net-list traversal, multiple word simulation has been incorporated in BISTSIM. BISTSIM has also adopted the techniques of simulation IDs proposed in PROOFS and desensitizing unnecessary test patterns proposed in FSIM to further improve the simulation speed. The experimental results have shown that BISTSIM is faster than FSIM by about 2 to 3 times for random-resistant circuits which require a large amount of test patterns.

For the signature evaluation, two algorithms, bit-array computation and parallel-pattern sequential simulation have been presented. The bit-array computation compresses the output responses and state of MISR and the value of feedback state (coefficients) into bit array. The next state of MISR is then evaluated with parallel word operation on these bit arrays. Alternatively, parallel-pattern sequential simulation evaluates signature by treating MISR as a sequential circuit and incorporates parallel pattern simulation with multiple iterations. To reduce the number of iterations, simulation of gates is ordered based on the structure of MISR. The experimental results have shown that the resultant BISTSIM outperforms the well-known sequential fault simulator HOPE1.1 by an average speedup ratio of 10.

References

[1] J. A. Waicukauski, E. B. Eichelberger, D. O. Forlenza, E. Lindbloom and T. McCarthy, "Fault Simulation for Structured VLSI," *VLSI Systems Design*, Vol. 6, No. 12, pp. 20-32, Dec. 1985.

[2] M. Abramovici, P. R. Menon and D. T. Miller, "Critical Path Tracing - An Alternative to Fault Simulation," *Proc. 20th Design Automation Conference*, pp. 2-5, 1987.

[3] K. J. Antreich and M. H. Schulz, "Accelerated Fault Simulation and Fault Grading in Combinational Circuits," *IEEE Trans. on Computer-Aided Design*, Vol. CAD-6, No. 5, pp. 704-712, Sept. 1987.

[4] B. Underwood and J. Ferguson, "The Parallel-Test-Detect Fault Simulation Algorithm," *Proc. International Test Conference*, pp. 712-717, 1989.

[5] F. Maamari and J. Rajski, "A Method of Fault Simulation Based on Stem Region," *IEEE Trans. on Computer Aided Design*, Vol. 9, No. 2, pp. 212-220, Feb. 1990.

[6] F. Maamari and J. Rajski, "The Dynamic Reduction of Fault Simulation," *Proc. International Test Conference*, pp. 801-808, 1990.

[7] F. Maamari and J. R. Rajski, "The Dynamic Reduction of Fault Simulation," *IEEE Trans. CAD*, Vol. 12, No. 1, pp. 137-148, Jan. 1993.

[8] H. K. Lee and D. S Ha, "An Efficient, Forward Fault Simulation Algorithm Based on The Parallel Pattern Single Fault Propagation", *Proc. International Test Conference*, pp. 946-955, 1991.

[9] T. M. Niermann, W. T. Cheng and J. H. Patel, "PROOFS: A Fast, Memory Efficient Sequential Circuit Fault Simulator," *IEEE Trans. on Computer Aided Design*, Feb. 1992, Vol. 11, No 2. pp. 198-207.

[10] N. Gouders and R. Kaibel, "PARIS: A Parallel Pattern Fault Simulator for Synchronous Sequential Circuits," *Proc. Int. Conf. on Computer-Aided Design*, Nov. 1991, pp. 542-545.

[11] M. Mojtahedi and W. Geisselhardt, "New Methods for Parallel Pattern Fault Simulation for Synchronous Sequential Circuits," *Proc. Int. Conf. on Computer-Aided Design*, Nov. 1993, pp. 2-5.

[12] H. K. Lee and D. S. Ha, "New Methods of Improving Parallel Fault Simulation in Synchronous Sequential Circuits," *Proc. Int. Conf. on Computer-Aided Design*, pp. 10-17, Oct. 1993.

[13] D. K. Bhavsar, "Self-Testing by Polynomial Division," *Proc. IEEE Test Conference*, pp. 208-216, 1981.

[14] J. Hartmann and G. Kemnitz, "How to Do Weighted Random Testing for BIST?," *Proc. Int. Conf. on Computer-Aided Design*, pp. 586-571, 1993.

[15] I. Pomeranz and S. M. Reddy, "3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits," *IEEE Trans. on Computer-Aided Designed,"* Vol. 12, No. 7, pp. 1050-1058, July 1993.

[16] S. Venkataraman, J. Rajski, S. Hellebrand and S. Tarnick, "An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers," *Proc. of Int. Conf. on Computer-Aided Design*, pp. 572-577, 1993.

[17] J. Hartmann, "On Numerical Weight Optimization for Random Testing," *Proc. of the European Conf. on Design Automation*, pp. 223-230, 1993.

[18] M. Bershteyn, "Calculation of Multiple sets of Weights for Weighted Random Testing," *Proc. of Int. Test Conf.*, Paper 45.3, pp. 1031-1040, 1993.

[19] P. H. Bardell, W. H. Mcanney and J. Savir, *Built-In Self Test for VLSI: Pseudorandom Techniques*, Jonh Wiley & Sons, New York, 1987.

[20] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinatorial benchmark circuits and a target translator in fortran," *Proceedings of International Symposium on Circuits and Systems*, June 1985.

[21] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Circuits," *Proc. International Symposium of Circuits and System* , May 1989, pp. 1929-1934.