PROOFS: Sequential Circuit Fault Simulator

Ramraj Gottiparthy, student member, IEEE
Srinivas Garimella(Reviewer)
Auburn University

Abstract **— PROOFS (Parallel RestOrative Order-independent Fault Simulator) is a hybrid of concurrent, differential, and parallel fault simulation algorithms. It retains fault dropping advantage of concurrent method, word level parallelism of parallel method and low memory requirement of differential, while minimizing their individual disadvantages. Parallel simulation of faults reduces simulation time of PROOFS. Reduction of faults to be simulated in parallel, efficient fault injection and efficient fault ordering to reduce the number of gate evaluations are the major factors behind success of this fault simulator. Fault ordering is crucial in exploiting the benefits of parallelism. Experimental results show that fault ordering by depth first search of the circuit starting at the primary outputs achieves 40% reduction of gate evaluations over a random fault order. Unexcited fault elimination by two levels achieved 45% simulation time reduction. Results of [1] demonstrate that parallel fault simulation using 32 bit word instead of 1 bit speeds up fault simulation by six times for the largest circuit. A comparison of PROOFS with Concurrent fault simulation shows that PROOFS is 6 to 67 times faster and requires only one fifth the memory required for concurrent fault simulation on the ISCAS-89 sequential benchmark circuits. The above results obtained are based on zero delay model of gates.**

INTRODUCTION

The complexity of VLSI circuits has been growing significantly throughout the last years. As circuits become larger, the design software tools are required to be more powerful to do all the required tasks efficiently. The software tool under study, fault simulator, is used for the development of manufacturing test. The fault simulator performs two functions[2]:

1.It determines the coverage of a given set of input vectors for a given fault model or a given fault list.

2.With the help of other programs(a test generator or vector compacter) it can produce a set of vectors with a given fault coverage for manufacturing test.

In addition, fault simulators are used to find unproductive (vectors which can be compacted with out reducing fault coverage)test vectors and dropping of such vectors speeds up test generation process. These fault simulators are used for generating fault dictionaries and for computing aliases in signature analysis. To perform all these operations for complex VLSI circuits require fault simulators that are fast, and use memory efficiently.

A number of fault simulators are developed based on single stuck at fault model. Table I represents task of fault simulation.

Table 1. The Tasks of Fault Simulation

	V_1	...	V_i	V_{i+1}	...	V_n
Good	G_1	...	G_i	G_{i+1}	...	G_n
Bad_1	$B_{1,1}$	...	$B_{1,i}$	$B_{1,i+1}$	...	$B_{1,n}$
Bad_2	$B_{2,1}$	...	$B_{2,i}$	$B_{2,i+1}$	...	$B_{2,n}$
.	.	...	.	.	...	.
Bad_k	$B_{k,1}$	...	$B_{k,i}$	$B_{k,i+1}$	...	$B_{k,n}$
Bad_{k+1}	$B_{k+1,1}$	...	$B_{k+1,i}$	$B_{k+1,i+1}$	...	$B_{k+1,n}$
.	.	...	.	.	...	.
Bad_m	$B_{m,1}$	...	$B_{m,i}$	$B_{m,i+1}$	...	$B_{m,n}$

There are m faulty machines and the test sequence has n vectors. Each column corresponds to a test vector and each row corresponds to good machine or one of the bad machines. The task of fault simulation is to generate the primary output values for each one of the (m+1)n machine status and determine which faulty machines have output vectors different from the good machine. Serial fault simulation, Parallel fault simulation, deductive fault simulation and concurrent fault simulation methods generate each machine status from its left neighbor from the same row [2]. Concurrent fault simulator is fastest but every line in the circuit is associated with a list, which requires large memory. Deductive fault simulation requires special set of operations, which have a large time overhead. Parallel simulation, simulates 32 (for a 32 bit machine) faulty machines per pass but suffers from inability to drop detected faults. Differential simulation determines $B_{i,j}$ from the state $B_{i-1,j}$. This simulation requires less memory but suffers from the inability to drop detected faults easily because subsequent faulty machines rely on differences from previous faulty machines. Single fault propagation generates each machine state from its reference machine state in the same column, however restoring the state of the good machine before every faulty machine simulations results in a performance overhead. Single fault propagation with word level parallelism is parallel pattern single fault propagation technique. Efficient heuristics were proposed to trace the fault effects in combinational circuits, which will not be useful for sequential circuits. All the above-mentioned drawbacks of different strategies are overcome by PROOFS. For example, fault dropping is easy in PROOFS, it stores the faulty machine values at the state nodes and it fully utilizes all bits in the word and avoids

resimulation of the good circuit by dynamic fault grouping strategy.

PROOFS ALGORITHM

(a) Definitions:

Inactive fault: If the stuck-at value of a single event fault f and the good value of the faulty line are identical, the fault f is not activated. Hence, it does not need to simulate the fault. In PROOFS, these kinds of faults, called inactive faults, are not injected for parallel fault simulation. The concept of inactive faults has been further extended, in the later version of PROOFS. If a single event fault f is active, the fault f is attempted to propagate one (or two) level further. If the fault does not propagate to any gate in the next one (or two) level, the fault f is not injected. A method, so called star algorithm, is also incorporated in advances versions. By incorporating these methods, the performance of PROOFS has been increased by, on the average, 17% for benchmark circuits [3].

Fault dropping: A fault is said to be dropped if that fault is detected and removed from current list of faults. Fault dropping reduces gate evaluations and speeds up simulation. If a fault is not detected, then all the state nodes with values different from the good machine values are stored in the linked list associated with the fault for the next vector.

Fault Ordering (Fault grouping): Fault grouping strategy regroups the remaining faulty machines into groups of 32 machines for each vector that is applied to the circuit. In order to reduce the overall processing time, faults that cause the same events should be grouped into the same packet. In PROOFS, faults are ordered by a depth first search from primary outputs towards primary inputs during processing time.

Fault Injection: Every fault in the fault group must be injected into the circuit. In PROOFS, s-a-1 fault is injected by using an extra OR gate in the circuit and s-a-0 is injected by using an extra AND gate in the circuit. As shown in the Figure.[1] s-a-1 fault at A in 2nd bit position is introduced by inserting OR gate at line A with other input of OR gate connected to a dummy gate whose input is all zeros except 2nd bit position.

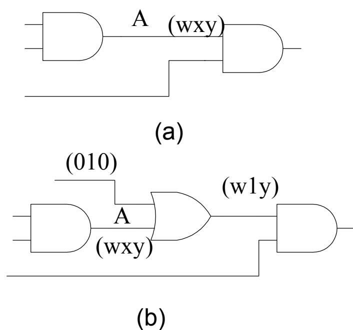


Figure 1.Fault Injection. (a) Before fault injection
(b) Circuit with line A s-a-1 injected at bit position 2

Group id: During the generation of every faulty machine status a unique simulation ID is given, which is called as group id. In order to restore the good machine status before every faulty machine simulation requires three different copies of the circuit status simultaneously. The first copy always stores good machine status. If any value is changed from good machine value, the new value is recorded in the second copy and group id is marked in third copy. *If the group id of line is same as current simulation id, the value of the line is chosen from second copy, otherwise its value is taken from first copy.*

(b)Data Structure: Four valued (0,1,X,Z) logic is used in PROOFS. Two bits are used to code the four values. 0 is coded as (1,0), 1 as (0,1), X as (0,0), and Z as (1,1). Table II shows the logic used to evaluate different gates, 32 faulty machines at a time.

TABLE II
GATE EVALUATIONS

	V0	V1
AND	$A_0 \mid B_0$	$A_1 \& B_1$
OR	$A_0 \& B_0$	$A_1 \mid B_1$
INV	A_1	A_0
XOR	$(A_0 \& B_0) \mid (A_1 \& B_1)$	$(A_0 \& B_1) \mid (A_1 \& B_0)$
TRIG	$E_0 \mid (A_0 \& E_1)$	$E_0 \mid (A_1 \& E_1)$
BUS	$(A_0 \& B_0 \& B_1) \mid (A_0 \& A_1 \& B_0)$	$(A_1 \& B_0 \& B_1) \mid (A_0 \& A_1 \& B_1)$

|: bitwise OR.
&: bitwise AND.

Good machine, faulty machine values consists of two 32-bit words, V0 and V1, where each bit is used to store a different faulty machine's value. Group id is stored in 32 bit word. This allows more than 4 billion faulty machine simulations without clean up.

(c) Algorithm:

A synchronous sequential circuit can be decomposed into flip-flops(FFs) and combinational logic blocks (CLBs). The removal of FFs turns a sequential circuit into a combinational circuit. The outputs of FFs are called pseudo-primary inputs(PPIs) and inputs of FFs pseudo-primary outputs(PPOs). Under the zero gate delay model, the sequrntial behavior of the circuit can be simulated by repeating the simulation of the CLBs at each time frame. After simulation of one group of faults, the PPOs which have different values as compared to good circuit are stored. The states of PPOs become PPIs in next time frame. The complete algorithm of PROOFS is shown in the Fig.2.The main loop which reads the next input vector, does the good machine evaluation. Group id is incremented and a packet of 32 faults are injected together with previous vector state node events and simulated using bit parallelism of a computer word for that time frame. Faults detected at POs are dropped, and state node events are stored for next input vector. This is repeated until all the faults are simulated. Then the next vector is taken, group id is incremented and above steps are repeated. Good fault grouping strategy simulates all the faults in minimum number of passes thus reducing the iteration count in the inner loop and speeds up the simulation. An example circuit is shown in the Fig.3. It is Combinational part of the circuit with equivalent collapsed fault list. Test vectors required to propagate these faults (A s-a-0, A s-a-1,B s-a-1,D s-a-1,E s-a-0,C s-a-0) to the output are 110,010,100,000,001. Two 6-bit

words are used to represent good circuit and bad circuit values. This structure accommodates all faults in single pass. A single pass with 110 input vector and faults detected by this vector are shown in the Fig.4.

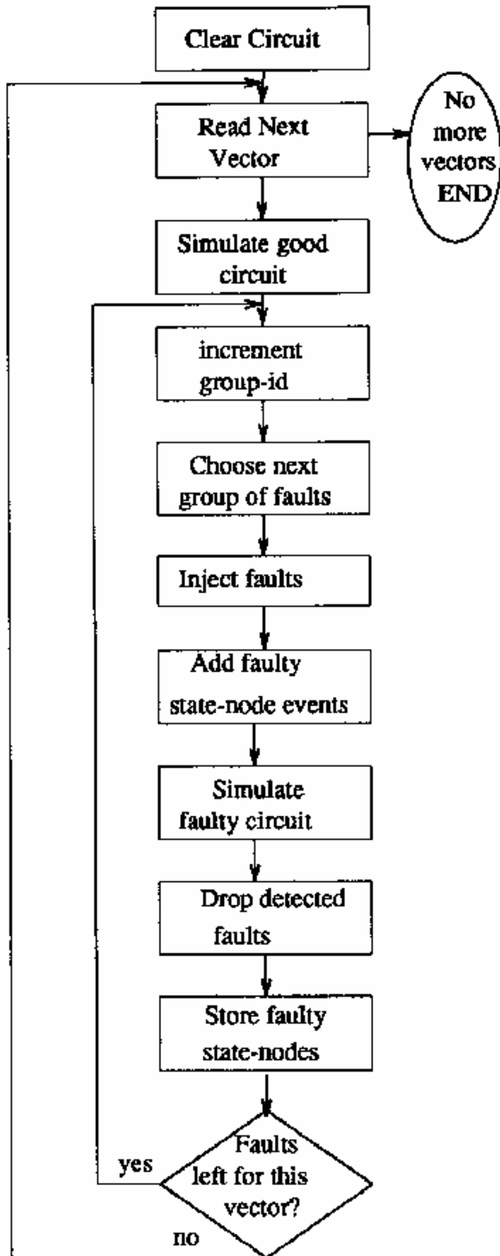


Figure 2. PROOFS algorithm.

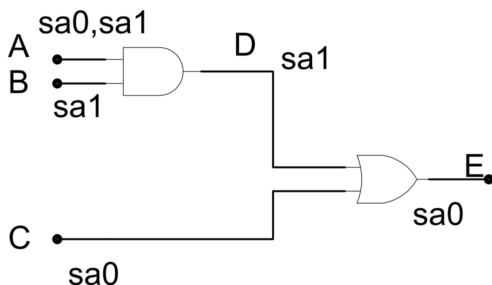
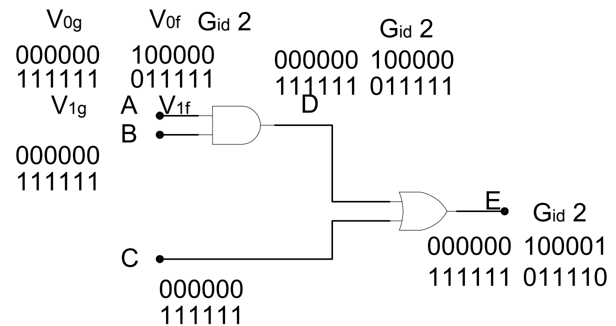


Figure 3. Circuit to illustrate PROOFS algorithm

V_{0g}, V_{1g} corresponds to words of good circuit values and V_{0f}, V_{1f} corresponds to words of faulty circuit values. Group id is represented using Gid. Current group id is taken as 2. Boolean operations are performed on input vectors using Table II. Whenever value of line differs from good circuit value, a second copy is made with that different value and group id is assigned for it. At the end of simulation, it is observed that bit 1 and bit 6 differs from good circuit values. Hence, faults Asa0, Csa0 are detected and these faults are dropped. Next input vector 010 is taken and simulation is run with incremented group id. This process repeated for all vectors.

Asao	Asa1	Bsa1	Dsa1	Esa0	Csa0

(a)



(b)

Figure 4.(a)Word structure (b)Faulty circuit simulation

(d)Applications and Limitations: PROOFS is applicable to all gate level circuits, which are functionally insensitive to the relative delays of gates. PROOFS assumes zero delay model for gates and this can easily extensible to unit delay simulation at the expense of more gate evaluations. PROOFS cannot be used for variable delay simulation, since every node in the circuit is potentially a state node at one time or another. It is not suitable for switch level fault simulation.

(e)Improvements: Many methods are investigated to reduce events in sequential circuit simulation by reducing the number of faults simulated for each test vector in [5], which requires identification of inactive faults in the circuit. PROOFS uses zero, one and two level inactive fault identification methods. Inactive faults are identified using local information from fault free circuit and star algorithm provides global information about inactive faults in sequential circuits. PROOFS implements static fault grouping method based on depth first search of the circuit starting at the primary outputs. Three algorithms [6] are proposed based on dynamic fault grouping during simulation and these algorithms are incorporated in PROOFS. These methods speed up simulation time by 55% for s35935 benchmark circuit. All these algorithms also showed at least 39% reduction in the number of faulty circuit gate evaluations.

CONCLUSION

PROOFS is the better fault simulator when compared to its earlier fault simulators like differential fault simulator and concurrent fault simulator in terms of speed and memory usage. Although new fault simulators based on parallel pattern single fault propagation like PARIS perform better than PROOFS in terms of simulation time, PROOFS is faster for large circuits. Fault simulator, HOPE, uses the idea of screening out of faults with short propagation paths through single fault propagation. HOPE is two times faster than PROOFS for most of benchmark circuits. Effective fault injection methods, better dynamic fault grouping strategies and taking advantages of PARIS at the expense of memory are some directions for the improvement of PROOFS fault simulator.

REFERENCES

[1] Thomas M. Neirmann, Wu-Tung Cheng and Janak H. Patel, *PROOFS: A fast, memory-efficient sequential circuit fault simulator,* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Feb. 1992,Volume: 11 , Issue: 2.

[2] Michael L. Bushnell, Vishwani D. Agrawal, *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI circuits*, Kluwar Academic Publishers, 2000, 80-120.

[3] Hyung Ki Lee and Dong Sam Ha, *New methods of Improving Parallel Fault Simulation in Synchronous Sequential Circuits,* IEEE/ACM International Conference, Nov.1993.

[4] Wu-Tung Cheng and Janak H. Patel, *PROOFS: A super fast simulator for sequential circuits,* Design Automation Conference, Mar. 1990.

[5] Elizabeth M. Rudnick, Thomas M. Neirmann and Janak H. Patel, *Methods for reducing events in sequential circuit fault simulation*, IEEE international conference on Computer-Aided Design, Nov. 1991.

[6] Charles R. Graham, Elizabeth M. Rudnick and Janak H. Patel, *Dynamic fault grouping for PROOFS: A win for large sequential circuits,* Tenth International Conference on VLSI Design, Jan. 1997.