

HyHOPE : A Fast Fault Simulator with Efficient Simulation of Hypertrophic Faults[†]

Chen-Pin Kung & Chen-Shang Lin

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abstract

In sequential circuit fault simulation, the *hypertrophic* faults, which result from lengthened initialization sequence in the faulty circuits, usually produce a large number of fault events during simulation and require excessive gate evaluations. These faults degrade the performance of fault simulators attempting to simulate them exactly. In this paper, an exact simulation algorithm is developed to identify the hypertrophic faults and to minimize their effects during the fault simulation. The simulator HyHOPE based on this algorithm shows that the average speedup ratio over HOPE 1.1 is 1.57 for ISCAS89 benchmark circuits. Furthermore, the result indicates the performance of HyHOPE is close to the approximate simulator in which faults are simply dropped when they become potentially detected.

1 Introduction

Fault simulation has been playing a major role in VLSI testing. Its applications range from grading the quality of test sets to incorporating with ATPG in test generation. As the size of VLSI circuits grow increasingly larger, efficient fault simulation algorithms have been developed to meet the challenge[1-12]. These algorithms are highly refined for their domains of applications. In this paper, we mainly concern with the fault simulation algorithms for single stuck-at faults of synchronous sequential circuits.

Most recently developed fault simulation algorithms are based on ROOFS [2]. ROOFS improves the performance of single fault propagation[1] by using an efficient circuit status restoration technique. Based on ROOFS, various parallelization techniques have been proposed to speed up the performance. Two fault simulators endeavor to parallelize the test vectors: PSF[6] and PARIS[7]. The major difference between PSF and PARIS is their ways of grouping test vectors into a packet, a computer word. In PSF, the test sequence is partitioned into consecutive subsequences, while a packet in PARIS represents consecutive test vectors. Both PSF and PARIS can achieve significant speedup over ROOFS, although their performance is somewhat correlated with the circuit types.

PROOFS[2-3] is the parallel-fault enhancement of ROOFS. In PROOFS, a packet of 32 active faults are injected and simulated parallelly. And its performance is further speeded up with fault ordering and efficient fault injection. A more efficient parallel fault simulator, HOPE is proposed in [4]. In HOPE, a single event fault in a fanout free region is simulated to stem with single fault propagation. And the stem fault is further examined by candidacy test. The results in [4] show that on the average, 67% of faults are screened out comparing with PROOFS. A new version of HOPE, HOPE 1.1[5] incorporates additional heuristics for further improvement.

Despite the above sophisticated techniques, the efficiency of the above fault simulation algorithms all depend on the degree of difference between the good circuit and the fault circuit. For

most faults, the fault effects are small in number and these simulators attain high efficiency by evaluating only a small number of gates for each fault. However, there are usually some hypertrophic faults in a circuit. These faulty circuits take longer sequences and are difficult to initialize. Hence, while good circuit is initialized, these faults produce a great number of unknown values (X's). As a result, a simulator taking advantage of such difference winds up heavily loaded with a large number of gate evaluations for the hypertrophic faults. In HOPE 1.1, almost half of fault events are from such hypertrophic faults using STG3[13] sequences on ISCAS'89 benchmarks and its performance greatly suffers from these faults. This phenomenon has long been known and the traditional way of dealing with hypertrophic faults is to drop a fault when it is potentially detected. Accuracy is thus sacrificed for higher simulation speed.

In this paper, we propose a novel and exact fault simulation algorithm to identify the hypertrophic faults during fault simulation and to efficiently simulate hypertrophic faults for sequential circuits such that their adverse effect on performance is minimized. Based on the proposed simulation algorithm, the reduction of gate evaluations resulting from hypertrophic faults is three folds:

- (1) The algorithm performs gate evaluation only when there is difference from their previous time frame rather than from the good circuit.
- (2) Each fault is simulated parallelly with logic simulation.
- (3) And the faulty circuits of various hypertrophic faults are simulated in parallel for even further reduction.

From this algorithm, a fault simulator, HyHOPE, is implemented upon the framework of HOPE 1.1. The experimental results show that HyHOPE reduces about **40%** fault gate evaluations of HOPE 1.1 and the average speedup ratio over HOPE 1.1 is 1.57 for ISCAS89 benchmark circuits. Furthermore, the results indicate that the performance of HyHOPE is close to an approximate version of HOPE 1.1 in which potentially detected faults are dropped.

The organization of this paper is as follows. In section 2, the characteristics of hypertrophic faults and the key observation for reducing hypertrophic fault effect will be discussed. In section 3, the simulation algorithm for hypertrophic faults in HyHOPE and its implementation will be described. A qualitative analysis on gate evaluations to demonstrate the advantage of HyHOPE will be given in section 4. In section 5, the performance of HyHOPE will be compared with HOPE 1.1. Finally, the conclusions will be given in section 6.

2 Hypertrophic Faults

In this section, the behavior of hypertrophic faults and their identification will be described. And the key observation for reducing the hypertrophic fault effect will be discussed.

2.1 Hypertrophic Faults

The hypertrophic faults[9] are faults which lengthen the initialization of the faulty circuits and during this period, cause the status of many gates in the faulty circuits to remain unknown, X's,

[†]This work was supported in part by the National Science Council under Contract NO. NSC-83-0404-E-002-055.

while good circuit has been initialized. In general, such hypertrophic faults of a circuit are small in number and are usually associated with control lines such as faults on reset lines. However, their fault effects spread widely at each time-frame and generate a great number of fault events. As a result, fault simulation for these faults requires large number of gate evaluations such that the performance of fault simulator is significantly degraded. The degradation in performance stems from the fact that modern fault simulators count precisely on the small difference between the good circuit and its faulty versions. This difference is small for non-hypertrophic faults which are the majority, and these fault simulators achieve good efficiency. The opposite characteristic of hypertrophic faults seriously affects the attainable efficiency of these simulators.

To handle hypertrophic faults, two approaches have been taken in the literatures. One approach is to handle these faults implicitly through either dynamic fault ordering [10] or dynamic fault grouping [5]. Both manipulate the ordering or group of faults during simulation in order to reduce the fault events. This approach is intuitive but the improvement is not significant. The results of [5] show that, the number of fault events can only be reduced about 17% for two of benchmark circuits.

Another approach is to relax the detection condition from sure detection to potential detection. A fault is surely detected if the fault produce complementary value of the good circuit at a primary output. And a fault is potentially detected if there exists a primary output whose faulty circuit value is unknown and the good circuit value is known. Since most of the hypertrophic faults are also potentially detectable. Hence, if a fault is dropped when it is potentially detected, the simulation time will be significantly reduced. Once dropped, a potentially detected fault remains unsure of its detectability even though some of them still have the chance to be surely detected in the later time frames. Accuracy is thus traded off for fewer simulation time. The simulation result is only approximate rather than exact.

We will propose an exact fault simulator which minimizes the simulation requirement for hypertrophic faults. First, the identification of hypertrophic faults will be described.

2.2 Identification of Hypertrophic Faults

The first step of handling hypertrophic faults is to identify these faults during the simulation. There have been a few methods to identify a hypertrophic fault. It can be either simply a fault has been potentially detected or as in Mozart [9], a fault which has the size of difference from the good circuit comparable to the logic events. These methods are more suitable for the approximate fault simulation in which these identified faults are dropped.

To simulate the hypertrophic faults exactly without incurring much overhead, these faults should be identified based on the estimation of the number of X's at the next time frame. The number of flip-flops at which the faulty circuit values are unknown and different from the good circuit is a good estimate for such purpose. Since with more X's at FFs, it is more likely that the difference from good circuit at the next time frame will be larger. If such count of a fault is large than a predetermined threshold value, the fault is identified as hypertrophic. The lower the threshold value, the sooner a hypertrophic fault is identified. However, there is a risk that some faults will be identified incorrectly as hypertrophic. Hence, the threshold value is a tradeoff between these two extreme. We had made a lots of experiments to choose a good threshold value. And in our implementation, the threshold value is determined as 5% of the number of flip-flops in a circuit. The overhead for identification is negligible comparing with the entire simulation time.

2.3 Key Observation

Once the hypertrophic faults are identified, an effective simulation algorithm will be developed to simulate them exactly. The simulation algorithm is based on the following observation.

In event driven simulation algorithms, logic or fault simulation, gate evaluation is required for a gate if there is any event occurs in the gate inputs. In other words, gate evaluation is required

if the gate input status is different from the reference circuit status. The reference circuit can be the same circuit in the previous time frame as in logic simulation, or the good circuit in the same time frame as in ROOFS. Hence, a fault simulation algorithms is more efficient if it has fewer fault events.

The following observation shows that the hypertrophic fault events can be reduced if a proper reference circuit is chosen.

Key Observation: *A hypertrophic fault tends to have less events with respect to the same faulty circuit in the previous time frame than to the good circuit. Furthermore these events are highly correlated with logic events of good circuit. In other words,*

$$|E_{f-g}| \gg |E_f - E_g|,$$

where E_{f-g} are the events corresponding to the difference between the faulty circuit and good circuit of current time frame, E_f are the events corresponding to the difference of faulty circuit between current time frame and the previous time frame, E_g are the logic events corresponding to the difference of good circuit between current time frame and the previous time frame, and the minus sign in $E_f - E_g$ is the set difference.

The validity of the observation can be demonstrated in Table 1 and will be further discussed in the gate evaluation analysis in section 4. In this table, 5 circuits with only hypertrophic faults are simulated with STG3 sequences. The hypertrophic faults are identified as described previously. For each circuit, $|E_{f-g}|$ and $|E_f - E_g|$ of all identified hypertrophic faults are listed in columns 4 and 5, and their ratio in the last column. The ratio clearly shows that $|E_f - E_g|$ is far less than $|E_{f-g}|$, less than 15% for all listed circuits.

Table 1. Hypertrophic fault events

circuit	#hyper.	$ E_g $	$ E_{f-g} $	$ E_f - E_g $	$\frac{ E_f - E_g }{ E_{f-g} }$
s382	22	73459	3757416	75813	0.02
s444	21	73349	3264580	71831	0.02
s526	27	36177	1519978	88769	0.06
s1488	7	255948	1393074	121877	0.09
s35932	10	751122	2970780	458875	0.15

From this key observation, an efficient simulation algorithm targeted for hypertrophic faults will be proposed in the next section.

3 HyHOPE

3.1 HyHOPE Algorithm

The above observation suggests that the hypertrophic faults should be simulated parallelly with logic simulation. Based on the key observation, the reduction of gate evaluations for hypertrophic faults is achieved by three ways:

- (1) Reduce the simulation loads of hypertrophic faults by simulating only the differences between two consecutive time frames of a faulty circuit instead of simulating its huge differences from the good circuit.
- (2) Combine the simulation of hypertrophic faults with logic simulation.
- (3) In addition, simulate the hypertrophic faults in parallel.

For non-hypertrophic faults, algorithms such as ROOFS-based algorithms are preferred, since these faults have only small difference from the good circuit. Therefore the HyHOPE algorithm is as Fig. 1.

The proposed algorithm is essentially to simulate the hypertrophic faults as in the classical parallel fault simulation[12] in which all faults are grouped into packets and each packet of faults are simulated parallelly with good circuit. The classical parallel fault simulation is not efficient because most faults are non-hypertrophic and produce less difference from good circuit than from the faulty circuits at previous time frame. *By identifying the hypertrophic faults, we are able to simulate both kind of faults, hypertrophic and non-hypertrophic, in their most efficient ways.*

```

All faults are in the regular fault list  $FL$ ;
For each test vector {
    simulate good circuit and hypertrophic faults parallelly;
    drop detected hypertrophic faults;
    while( there are faults  $\in FL$  not simulated for this vector) {
        simulate fault(s) with the ROOFS -based algorithm;
        identify hypertrophic faults;
        for each identified hypertrophic {
            mark the fault as hypertrophic fault;
            remove it from  $FL$ ;
        }
    }
}

```

Figure 1. The HyHOPE Algorithm

3.2 Implementation of HyHOPE

The proposed algorithm for simulating hypertrophic faults can be incorporated with existent fault simulators such as ROOFS and its parallel fault versions, PROOFS and HOPE1.X. We develop our simulator HyHOPE upon HOPE 1.1 because it has the best performance and its source code has been made publicly available by its authors.

In HyHOPE, procedures for hypertrophic fault identification and processing are built upon the original simulation mechanism of HOPE 1.1. Therefore, the proven techniques of HOPE and new methods incorporated in HOPE 1.1 are retained in HyHOPE for non-hypertrophic faults. All faults in the regular fault list are initially classified as non-hypertrophic. After having simulated a packet of faults, the hypertrophic-fault-identification procedure is invoked to identify hypertrophic faults. The identified faults are then removed from the regular fault list into a hypertrophic fault list. At the next time frame, faults in the hypertrophic fault list are simulated parallelly with the logic simulation. The original procedure for logic simulation is enhanced such that good circuit and faulty circuits of hypertrophic faults can be simulated simultaneously. A fault in hypertrophic fault list remains in the list till the fault is detected and dropped.

Accordingly, some data structures in HOPE 1.1 are also modified. In HyHOPE, an extra word pair for each gate is added to parallelly record the good circuit status and hypertrophic faulty circuit status. This word pair is evaluated during the parallel simulation of good and hypertrophic faulty circuits. The first bit of word pair is used to represent the good circuit status. And after the parallel simulation, its value is copied to original computer words representing good circuit status. The original good-circuit words are used for both parallel fault simulation and single fault propagation of non-hypertrophic faults as in HOPE 1.1. It is possible to remove the original good-circuit words since in HyHOPE there is already a bit pair representing the good circuit status. However, the good circuit status then has to be retrieved during non-hypertrophic fault simulation and more overhead will be introduced in each gate evaluation. Since the number of logic events is generally less than the number of fault events in sequential circuit fault simulation, we eliminate this overhead with small increase in memory space. For experimented circuits, the memory requirement of HyHOPE is larger than HOPE1.1 by only 6% on average.

In the present implementation of HyHOPE, the maximum number of hypertrophic faults to be simulated parallelly with logic simulation is limited by the given computer word length, 32 minus one for good circuit simulation. The number limitation does not affect the performance of HyHOPE in the evaluation. Since the number of hypertrophic faults is small in general and the space is reused when some identified hypertrophic faults are detected. We would also like to point out that the hypertrophic fault packet size can be extended to more than one word, and more hypertrophic faults can be simulated simultaneously.

4 Gate Evaluation Analysis

In this section, the efficiency of various fault simulation algorithms will be discussed based on the analysis of the cases for which gate evaluation must be performed.

In simulation, a gate evaluation may occur when a value transition of a gate input between two consecutive time frames or a value difference between good and faulty circuits takes place. Table 2 shows all the cases for the changing status at a gate input between consecutive two time frame T_{i-1} and T_i for a fault F . These cases are enumerated according to four following conditions:

1. the faulty circuit status F_{i-1} is different from the good circuit status G_{i-1} at T_{i-1} ,
2. the good circuit status is changed from T_{i-1} to T_i ,
3. the faulty circuit status is changed from T_{i-1} to T_i and
4. the faulty circuit status F_i is different from the good circuit status G_i at T_i .

A datum of value 1 in Table 2 indicates that a gate input meets the condition, otherwise, the value 0 is given. It is easy to see that four of the cases are not possible and C0, C6 and C9 result in no gate evaluation in a worthy fault simulator.

For a given simulation algorithm, a case may or may not result in gate evaluation and these characteristics determine the efficiency of the simulator. The simulation algorithms of ROOFS, concurrent fault simulation[11] and HyHOPE for hypertrophic faults are considered in this section. Note that the discussion on ROOFS can also be applicable to its various parallelized versions: PROOFS and HOPE1.X because the parallelization of faults can only reduce but not completely eliminate the gate evaluations in cases of Table 2.

Table 2. Cases of gate evaluation

Case	Gate input status				Examples
	$F_{i-1} \neq G_{i-1}$	$G_{i-1} \neq G_i$	$F_{i-1} \neq F_i$	$F_i \neq G_i$	
C0	0	0	0	0	0, x
C1	0	0	0	1	impossible
C2	0	0	1	0	impossible
C3	0	0	1	1	$1 \rightarrow 1/0, x \rightarrow x/0$
C4	0	1	0	0	impossible
C5	0	1	0	1	$1 \rightarrow 0/1, x \rightarrow 0/x$
C6	0	1	1	0	$0 \rightarrow 1, 1 \rightarrow 0$
C7	0	1	1	1	$x \rightarrow 1/0, 0 \rightarrow 1/x$
C8	1	0	0	0	impossible
C9	1	0	0	1	$1/0, 1/x$
C10	1	0	1	0	$1/0 \rightarrow 1, 1/x \rightarrow 1$
C11	1	0	1	1	$1/x \rightarrow 1/0, 1/0 \rightarrow 1/x$
C12	1	1	0	0	$1/0 \rightarrow 0, 1/x \rightarrow x$
C13	1	1	0	1	$1/x \rightarrow 0/x, x/0 \rightarrow 1/0$
C14	1	1	1	0	$1/0 \rightarrow x, 1/x \rightarrow 0$
C15	1	1	1	1	$1/0 \rightarrow 0/1, 1/x \rightarrow 0/1$

In Fig. 2, the cases for which a simulation algorithm must perform gate evaluation are shown in the shaded area of the gate evaluation map. For example, a gate is evaluated in ROOFS for a fault if the gate has a gate input falls into any of the cases C3, C5, C7, C9, C11, C13, and C15. And in concurrent fault simulation, gate evaluation is performed for a fault if any of the cases C3, C5, C7, C10, C11, C12, C13, C14, C15, and C6-9 occurs, where C6-9 is the case that a gate has inputs satisfying both conditions of C6 and C9. This is the case that a fault is in the fault list of the gate and there are also logic events at gate inputs.

For both ROOFS and concurrent fault simulation, the cases indicated in their respective maps are associated with gate evaluations independent of those of logic simulation i.e., the gate evaluations of faults resulting from these cases must be performed in addition to those of Fig.2a. On the other hand, for the hypertrophic faults in HyHOPE, they are simulated parallelly with logic simulation and hence, part of their gate evaluation cases coincide with those of logic simulation, as indicated in Fig.2d. Therefore, only the non-coincident cases, C3, C10, and C11, result in additional gate evaluations. As a result, it can be concluded

that HyHOPE is superior to concurrent fault simulation in dealing with hypertrophic faults because the shaded area of the latter covers more than C3, C10, and C11, and thus needs more gate evaluations than HyHOPE.

The gate evaluation maps of ROOFS and HyHOPE show that for hypertrophic faults, ROOFS have additional gate evaluations for C5, C7, C9, C13 and C15 comparing with HyHOPE, while HyHOPE has additional evaluation for C10 comparing with ROOFS. From Table 2, it can be seen that C10 occurs when faulty circuit changes status while good circuit remain unchanged. For hypertrophic faults, such occurrence is likely to be far less than cases like C5, C9 and C13 for which faulty circuit is plagued by X's and good circuit has known status. As a result, HyHOPE can achieve better efficiency than ROOFS for hypertrophic faults. And since HyHOPE adopts same algorithm as ROOFS for non-hypertrophic faults, the overall performance of HyHOPE is significant better than ROOFS-based simulators as will be demonstrated experimentally in the next section.

C0	X	C3	X
X	C5	C7	C6
C12	C13	C15	C14
X	C9	C11	C10

(a) Logic simulation

C0	X	C3	X
X	C5	C7	C6
C12	C13	C15	C14
X	C9	C11	C10

(b) ROOFS (all faults)

C0	X	C3	X
X	C5	C7	C6
C12	C13	C15	C14
X	C9	C11	C10

(c) Concurrent fault simulation (all faults)

C0	X	C3	X
X	C5	C7	C6
C12	C13	C15	C14
X	C9	C11	C10

(d) HyHOPE (hypertrophic faults)




-  The darkly shaded area indicates the cases for which a simulation algorithm must perform gate evaluations.
 For concurrent fault simulation, in addition to the darkly shaded area, gate evaluation is required for the case C6-9 as indicated by the lightly shaded area in (c).
 The simulation for hypertrophic faults in HyHOPE is combined with the logic simulation as indicated by the dotted area in (d).

Figure 2. Gate evaluation maps

The above analysis indicates that the simulation algorithm for hypertrophic faults in HyHOPE is superior to the concurrent fault simulation and ROOFS. In the next section, the experimental result will be given to show HyHOPE indeed has better performance than HOPE 1.1, a ROOFS-based simulator.

5 Experimental Results

Our fault simulator HyHOPE is evaluated on ISCAS89 benchmark circuits[14] with test vectors generated by STG3. Throughout all the experiments, the initial states of all the flip-flops are assumed to be unknown, X's. Table 3 shows a summary of these circuits and test vectors. The performance of HyHOPE will be compared with those of HOPE 1.1 and PTD which is a modified version of HOPE 1.1 to drop potentially detected faults.

Table 3 also shows the fault coverages of exact fault simulation such as HyHOPE and HOPE 1.1 and the approximate fault simulation by PTD. As expected, PTD always gives an optimistic result. The number of hypertrophic faults identified and processed by HyHOPE is also given. For some circuits, this number is larger than 31, which means that during the simulation, some identified hypertrophic faults are detected and their bit spaces are reused in the following time frames. It can also be seen that the hypertrophic faults are generally less than 5% of the total faults. In

particular, the identified hypertrophic faults in s35932 are only 0.03%, yet a large number of gate evaluations are due to these small number of faults.

The run times of HyHOPE and HOPE 1.1 on a SUN4 Sparc2 workstation are reported in Table 4. The performance of HyHOPE can be clearly seen in this table. The average speedup ratio of HyHOPE over HOPE 1.1 is 1.57. As shown in the table, HyHOPE is faster than HOPE 1.1 for all but two circuits. For circuit s1238 and s953, HyHOPE is slightly slower than HOPE 1.1 because there are very few serious hypertrophic faults in these two circuits. The circuit s1238 has no feedback loop and it is very easy to be initialized. The circuit s953 is hard to be completely initialized, more than half of the flip-flops are not initialized during the simulation. Both these cases will not activate the hypertrophic fault effects and the overhead for hypertrophic fault identification and processing makes performance of HyHOPE be slightly slower than HOPE 1.1. It is also interesting to note that for circuits s382, s400 and s444, the speedup ratios over HOPE 1.1 are more than 2 while their identified hypertrophic faults are less than 6% of total faults. This indicates hypertrophic faults consume a large portion of the entire simulation time in HOPE 1.1 and the efficient algorithm of HyHOPE leads to a significant improvement.

For evaluation purpose, we also show the run time of the approximate fault simulator PTD in Table 4. Since PTD drops the potentially detected faults which are mostly hypertrophic faults, its performance can be served as a goal which an exact fault simulator of same basic simulation mechanism strives to achieve. As expected, PTD outspeeds its exact version HOPE 1.1 by 83% on average with some sacrifice in accuracy. When compared with HyHOPE, PTD is only slightly faster than the exact simulator HyHOPE. This indicates that HyHOPE not only preserves the accuracy but also has the performance approaching that of the approximate simulator.

To examine our proposed simulation algorithm for hypertrophic faults in detail, the numbers of gate evaluations for faulty circuit simulation are listed in Table 5. For HOPE 1.1 and PTD, the gate evaluations consist of evaluations for single fault propagation, candidacy test and parallel fault simulation. The listed gate evaluations of HyHOPE consist of two parts: the column **fault** in HyHOPE is the number of gate evaluations for faulty circuit generated by non-hypertrophic faults which are simulated as in HOPE 1.1; and the column **extra** lists the extra evaluations for simulating hypertrophic faults parallelly with logic simulation as proposed in our simulation algorithm. Therefore, the number of total gate evaluations for faulty circuit simulation with HyHOPE is the sum of these two numbers. From this table, it can be seen that about 40% gate evaluations for faulty circuits in HOPE 1.1 are reduced by HyHOPE on average. The reduction ratio is approaching to the ratio by PTD. Furthermore, there are many circuits such as s382, s444, s526, s832, s1488 and s35932, the efficiency of HyHOPE is very close to PTD.

In summary, the HyHOPE improves the performance of HOPE 1.1 by about 60%. Furthermore, the results show that with our efficient algorithm for simulating hypertrophic faults, *the exact fault simulator can be almost as fast as the approximate one, without any sacrifice in accuracy.*

6 Conclusion

In this paper, we have proposed a novel and exact fault simulation algorithm to identify the hypertrophic faults during the fault simulation and to efficiently simulate hypertrophic faults for sequential circuits such that their effect on performance is minimized. The reduction of hypertrophic fault events by the proposed simulation algorithm is three folds:

- (1) The algorithm reduces the simulation loads of hypertrophic faults by simulating only the differences between two consecutive time frames of a faulty circuit instead of simulating its huge differences from the good circuit.
- (2) Each hypertrophic fault is simulated parallelly with logic simulation. The events can be significantly re-

- Based on this algorithm, a fault simulator HyHOPE for efficient and exact hypertrophic fault simulation has been implemented upon HOPE 1.1. The experimental results have shown that HyHOPE reduces about **40%** gate evaluations of faulty circuit simulation from HOPE 1.1 and the average speedup ratio over HOPE 1.1 is **1.57**. Furthermore, the experiment results show that the exact fault simulator based on our efficient algorithm for simulating hypertrophic faults can be almost as fast as the approximate one, without any sacrifice in accuracy.

We would like to thank Dr. Dong Sam Ha for providing the source code of HOPE 1.1.

- [1] F. Ozguner, et al., "On Fault Simulation Techniques," *Journal of Design Automation and Fault Tolerant Computing*, Vol. 3, No. 2, pp. 83-92, 1979.
- [2] W. T. Cheng and J. H. Patel, "PROOFS: A Super Fast Fault Simulator for Sequential Circuits," *Proc. The European Conference on Design Automation*, pp. 475-479, 1990
- [3] 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*, Vol. 11, No 2. pp. 198-207, Feb. 1992.
- [4] H. K. Lee and D. S. Ha, "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits," *Proc. 29th Design Automation Conference*, pp. 336-340, June 1992.
- [5] 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.
- [6] C. P. Kung and C. S. Lin, "Parallel Sequence Fault Simulation for Synchronous Sequential Circuits," *Proc. The European Conference on Design Automation*, pp. 434-438, Mar. 1992.
- [7] N. Gouders and R. Kaibel, "PARIS: A Parallel Pattern Fault Simulator for Synchronous Sequential Circuits," *Proc. Int. Conf. on Computer-Aided Design*, pp. 542-545, Nov. 1991.

Circuits	No. faults	No. tests	No. hyp. faults	Fault coverage (%)	
				sure	potential
				HOPE 1.1, HyHOPE	PTD
s208	215	111	28	63.72	72.09
s298	308	162	15	85.71	88.64
s344	342	91	30	96.20	97.95
s382	399	2463	22	90.98	94.74
s400	424	1282	21	82.78	86.56
s420	430	173	41	41.63	50.70
s444	474	1881	21	89.45	92.62
s526	555	754	27	75.32	78.38
s641	467	133	20	86.30	87.79
s713	581	107	43	80.90	83.13
s820	850	411	5	81.88	82.12
s832	870	377	5	81.38	81.49
s838	857	137	37	29.64	38.39
s953	1079	16	32	7.78	15.01
s1238	1355	349	10	94.69	94.69
s1423	1515	36	51	24.42	28.12
s1488	1486	590	7	92.60	92.80
s1494	1506	469	7	91.10	91.43
s5378	4603	408	45	74.02	75.32
s35932	39094	86	10	87.99	88.04

Circuits	CPU time (sec.)			Speedup over HOPE 1.1	
	HOPE 1.1	PTD	HyHOPE	PTD	HyHOPE
s208	0.32	0.22	0.23	1.45	1.39
s298	0.43	0.23	0.28	1.87	1.54
s344	0.33	0.20	0.25	1.65	1.32
s382	8.17	1.87	2.08	4.37	3.93
s400	6.20	1.85	2.03	3.35	3.05
s420	1.12	0.63	0.83	1.78	1.35
s444	8.60	3.33	3.65	2.58	2.36
s526	5.50	2.87	3.07	1.92	1.79
s641	0.57	0.42	0.48	1.36	1.19
s713	0.57	0.42	0.52	1.36	1.10
s820	2.13	1.40	1.63	1.52	1.31
s832	2.02	1.37	1.53	1.47	1.32
s838	2.48	1.42	2.08	1.75	1.19
s953	0.70	0.67	0.73	1.04	0.96
s1238	1.87	1.83	1.97	1.02	0.95
s1423	1.67	1.50	1.60	1.11	1.04
s1488	5.83	3.03	3.68	1.92	1.58
s1494	4.92	2.65	3.23	1.86	1.52
s5378	21.12	12.27	15.62	1.72	1.35
s35932	64.00	44.30	50.92	1.44	1.26
Average speedup ratio				1.83	1.57

[illegible]