Improving Performance of Effect-Cause Diagnosis with Minimal Memory Overhead

Huaxing Tang*, Chen Liu†, Wu-Tung Cheng*, Sudahkar M. Reddy† and Wei Zou*
*Mentor Graphics Corporation, 8005 S.W. Boeckman Road, Wilsonville, Oregon, USA, 97070
†ECE Dept., University of Iowa, Iowa City, Iowa, USA, 52242

***Abstract*— Effect-cause diagnosis procedures are the most commonly used in industry to diagnose VLSI circuits that fail manufacturing test or field applications. Fast and effective diagnosis procedures are essential to diagnose large numbers of failing dies for yield ramp-up. We have recently proposed a method to speed up effect-cause diagnosis procedures by using a dictionary of small size [26]. In this paper we propose methods to further reduce the dictionary size and still achieve higher performance. Experiments on several industrial designs demonstrate that, on average, effect-cause diagnosis procedures can be speeded up by 3.5X while requiring minimal memory overhead for a very small dictionary.**

I. Introduction

When the chip fabrication process moves into deep sub-micron domain, feature-related systematic defects, such as single vias, bridges over wide metal, etc. cause more and more yield problems. In the future traditional yield learning approaches, such as test chips, memory bitmap, physical failure analysis, etc., may become less effective for yield learning. Hence new yield learning methods based on volume diagnosis results have gained a lot of attention recently [4], [11]–[14], [16], [19]. The basic idea is to use the manufactured dies as their own test chips. As shown in [20], both the accuracy and the throughput of diagnosis are critical for successful yield learning.

Diagnosis methods can be classified into two categories: cause-effect diagnosis and effect-cause diagnosis. Cause-effect diagnosis, also called dictionary based diagnosis, pre-computes failing responses of all modeled faults for a design and stores them in a dictionary. A main problem with cause-effect diagnosis is that a very large memory may be needed to store the pre-computed dictionary. Although a number of techniques [8], [10], [17] have been proposed to reduce the memory overhead of the dictionary, it can still be too large to be applicable for modern large designs with tens of millions of gates. Moreover the diagnosis accuracy and resolution may be impaired if the stored test response information is incomplete, such as in a pass/fail dictionary [17]. Instead of performing table lookup, the effect-cause diagnosis [1] finds initial candidates by backtracing from failing outputs, and simulating them to find the real suspects. Compared to the cause-effect diagnosis, effect-cause diagnosis procedures [1], [3], [5]–[7], [15], [21]–[25] typically can achieve higher accuracy and better resolution whilst requiring negligible memory overhead. However, effect-cause diagnosis requires a longer time, ranging from several seconds to hours, to diagnose a single failing die due to the need for a large number of fault simulations. Hence the effect-cause diagnosis may not be able to satisfy the throughput requirement of the volume diagnosis for large designs. A new method to speed up effect-cause diagnosis procedures by using a small dictionary was proposed in [26]. However for large industrial designs with tens of thousands of test patterns the dictionary proposed in [26] may still be too large to be applicable.

In this paper we propose a diagnosis algorithm which combines backtracing from failing outputs and fault dictionary to process failing patterns to find initial candidates in effect-cause diagnosis procedures. X-algorithm [2] is used to effectively handle failing patterns with many failing bits, as in conventional effect-cause diagnosis. A failing bit is defined as a failing observation point for a specific failing pattern. An improved pre-computed small dictionary is used to quickly find the initial candidates for failing patterns with a small number of failing bits. Two techniques are proposed to effectively reduce the memory overhead of the dictionary down to a negligible level, to ensure the applicability of the proposed diagnosis method to the largest designs. Experimental results on several industrial designs demonstrate that a significant speedup can be achieved with minimal memory overhead by the proposed algorithm.

The paper is structured as follows: In Section II, a review of effect-cause and cause-effect diagnosis procedures and the earlier procedure to speed up effect-cause diagnosis procedures are given. In Section III we present the proposed techniques to reduce the memory overhead of the proposed dictionary and also present the proposed diagnosis algorithm. Experimental results are given in Section IV. Section V concludes the paper.

II. Preliminaries

A. Review of Effect-Cause Diagnosis

The effect-cause diagnosis procedures are typically based on Single Location at a Time (SLAT) patterns [5]. A SLAT pattern is a test pattern that produced a failing response on the tester which can be matched (referred to as explained) by the response to the test pattern by the circuit under test with a single fault at some circuit node. These procedures are composed of three parts:

TABLE I

DESIGN INFORMATION AND DICTIONARY SIZE

	D1	D2	D3	D4	D5	D6	D7
N_{Gate}	314K	543K	1.1M	1.1M	2.0M	506K	1.3M
N_{ObsPt}	20K	46K	64K	70K	134K	13K	8K
N_{Pat}	5000	2252	1999	9415	1000	1000	1800
R_{Comp}	n/a	n/a	n/a	n/a	n/a	6.5X	9.9X
N_{SAF}	631K	1.1M	1.7M	1.8M	4.2M	817K	2.5M
S_{Full}	7.9T	14.3T	27.2T	147T	70.4T	1.3T	4.5T
$S_{P/F}$	394M	310M	425M	2.1G	525M	102M	563M

1) For each failing pattern P^i_{fail}, use the X-algorithm [2] from failing outputs to drop the undetectable faults and obtain the initial list of candidate fault sites. Fault simulate the initial candidates to filter out the candidates which can not explain P^i_{fail}.
2) Find a minimal set of candidates to explain as many failing patterns as possible by solving a minimum set covering problem. The selected candidates are called suspects and are charaterized (as stuck-at, open, bridge etc.) based on their excitation conditions [7], [25] .
3) Simulate all suspects under passing patterns and compute scores for each suspect candidate based on its failing pattern matches and passing pattern mismatches. The suspects are next ranked based on their scores and the ranked list of suspect candidates are reported out.

Typically, for large modern designs, test data compression techniques are necessary to address the increasing test data volumes. Additionally sequential test patterns are used to achieve higher fault coverage and detect timing-related defects. For such cases, it is observed that the first stage of the effect-cause diagnosis tends to be much more time-consuming and hence should be given a higher priority during performance optimization of defect diagnosis tools.

B. Review of Cause-Effect Diagnosis

Another approach for diagnosing fail logs is called cause-effect diagnosis, where a fault dictionary is pre-computed and stored for a specific design and a given set of test patterns. During diagnosis, a quick lookup of fault dictionary is performed to determine suspects, which can explain the targeted failing pattern(s). Additional analysis can be performed to find out passing pattern matches(mismatches) information, compute the suspect score and rank the suspect list. In the extreme case, where all needed information is stored and simulation is totally bypassed, the cause-effect diagnosis could be extremely fast because the table lookup can be done with little CPU time.

Table I reports information on seven industrial designs used in this work. We also include the sizes of dictionaries for single stuck-at faults. Sizes for two dictionaries referred to as $full$ and $pass/fail$ (P/F) are shown. For each design, the number of gates (N_{Gate}), the number of observation points (N_{ObsPt}) and the number of test patterns (N_{Pat}) are presented. For designs with test response compaction technique implemented, the compaction ratio (R_{Comp}) is also reported. The compaction ratio is defined as $R_{Comp} = N_{Chain}/N_{Channel}$, where N_{Chain} is the number of scan chains and $N_{Channel}$ is the number of scan channels. For designs without the compaction technique, like $D1$ to $D5$, all primary outputs and scan flip-flops are treated as observation points, thus $N_{ObsPt} = N_{FF}+N_{PO}$, where N_{FF} is the number of scan flip-flops and N_{PO} is the number of primary outputs. For designs with the compaction technique implemented, like $D6$ and $D7$, the number of observation points is computed as: $N_{ObsPt} = N_{Channel} * L_{LongestChain} + N_{PO}$, where $L_{LongestChain}$ is the length of the longest scan chain.

To store the complete information on test fails in a dictionary one could use a single bit to indicate whether a given fault F_i is detected by a pattern P_j at an observation point O_k or not, which results in a $full$ dictionary. The size of a full dictionay in bytes can be computed as: $S_{Full} = N_{SAF} * N_{Pat} * N_{ObsPt}/8$, where N_{SAF} is the total number of collapsed stuck-at faults of this design. The size of a full dictionary can be prohibitively high. As we can see from Table I, even for a small design like $D1$, a memory of $7.9 * 10^{12}$ bytes is needed which makes the cause-effect diagnosis based on full dictionaries impractical for large designs. To reduce the memory overhead of a fault dictionary, a simple approach is to use a single bit to indicate whether a given F_i is detected by a pattern P_j or not, which results a $pass/fail$ dictionary. The size of a pass/fail dictionary in bytes is reduced to: $S_{P/F} = N_{SAF} * N_{Pattern}/8$, by discarding the failing bit information for every failing pattern, which is more reasonable as shown in Table I. However, the accuracy and resolution of diagnosis results based on pass/fail dictionary are typically reduced and may be unacceptable for some applications, such as physical failure analysis. Furthermore, even the reduced size of a pass/fail dictionary may be too large for some designs.

C. Signature-based Small Dictionary [26]

A fault dictionary of small size is proposed in [26], where only unique failing responses for each fault are stored. Failing responses are compressed into 32-bit signatures to further reduce the memory overhead. Also information on the clocks activated during the capture of responses to failing patterns is used to skip simulation of some of the passing patterns. This method works well when the memory budget is not limited.

However, it is observed that the size of the small dictionary increases almost linearly with the number of test patterns, as shown in Fig. 1. Small dictionaries proposed in [26] were created for N-detect test sets of design $D1$ in Table I, $N = 1, 3, 5, 7, 10$. As the pattern count increases from $5K$ (1-detect) to $39K$ (10-detect), the number of entries to be stored for fault dictionary increases from $1.2 * 10^7$ to $5.0 * 10^7$, and there is no obvious sign of saturation. Such linear increase in memory

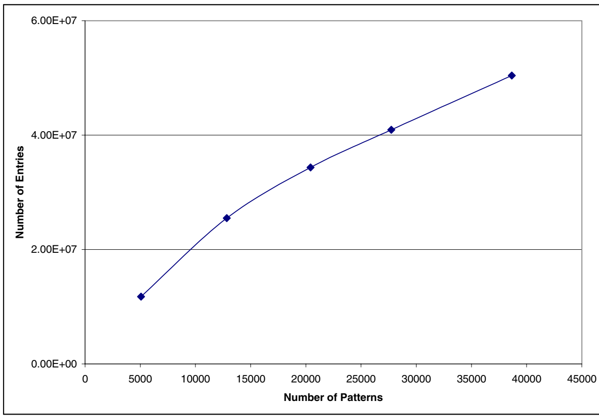

Fig. 1. Size of small dictionary [26] for $D1$

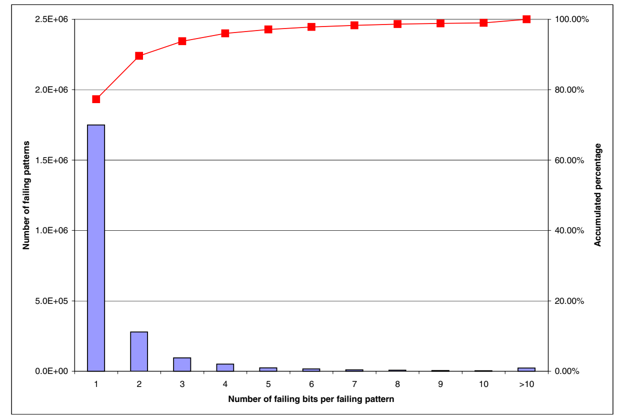

Fig. 2. Number of failing bit per failing pattern distribution

overhead with pattern counts caused the creation of the small dictionary to fail for a design with more than 10 millions gates and 27 thousand test patterns on a machine with $16GB$ random access physical memory. Thus for very large modern designs with tens of millions of gates and a large number of test patterns the memory overhead of diagnosis procedures must be minimized.

III. The Proposed Techniques and Diagnosis Procedure

In this section we propose methods to efficiently combine the advantages of cause-effect and effect-cause diagnosis procedures into an integrated diagnosis procedure that is highly efficient and requires minimal memory overhead. The same high diagnosis accurary and resolution as the effect-cause diagnosis is achieved by the proposed procedure. After discussing the proposed techniques to reduce the memory overhead of a fault dictionary, the proposed diagnosis procedure using the reduced size fault dictionary is presented. The observation which motivated us to investigate the methods to reduce dictionary size is the following. In Fig. 2 we plot the number of failing patterns with a given number of failing bits in tests that failed approximately $26,000$ different real chips during manufacturing test. The curve gives the cumulative percentage of all failing patterns and the X-axis represents the number of failing bits in the failing patterns. It can be seen that approximately 77% of the failing patterns have a single failing bit and over 98% of the failing patterns have 5 or less failing bits. Thus if one wants to improve the efficiency of effect-cause diagnosis procedures one should focus on improving the run time for most likely failing responses which contain very few failing bits as illustrated in Fig. 2. In this context we should also mention that the fact that test responses for a large percentage of faults in circuits reach only a small number of circuit outputs was also used to speed up diagnostic test generation and diagnostic fault simulation in [18].

A. N_{FB} Dictionary

We propose to use pre-computed failure information to handle failing patterns with few failing bits, which may not be effeciently processed by the X-algorithm [2] used in effect-cause diagnosis procedures. A fault dictionary is used to store all unique failing responses for each fault, with N_{FB} failing bits or less in a dictionary called the N_{FB} $dictionary$. When analyzing a failing pattern P^i_{fail} with no more than N_{FB} failing bits, a table lookup of the pre-computed N_{FB} dictionary is used to quickly find the initial candidate list for P^i_{fail}. The N_{FB} dictionary can be constructed in a preprocessing step as follows.

1) For each fault f_i, find the number of failing bits in responses by simulating all patterns.
2) Drop all failing patterns with more than N_{FB} failing bits.
3) Find the unique failing bit combinations for all remaining failing patterns of f_i.
4) Encode the unique failing bit combinations into 32-bit signatures and store them.
5) Repeat step (1) to (4) for all faults.
6) Reorganize unique signatures for failing patterns of all faults, such that they can be efficiently queried during diagnosis.

Since only the unique signatures for failing patterns with no more than N_{FB} failing bits are stored, the memory requirement is dramatically reduced, which makes the N_{FB} dictionary applicable to very large industrial designs.

B. Fault Grouping

Another technique is proposed to further reduce the memory usage of an N_{FB} dictionary. We give a sketch of this technique

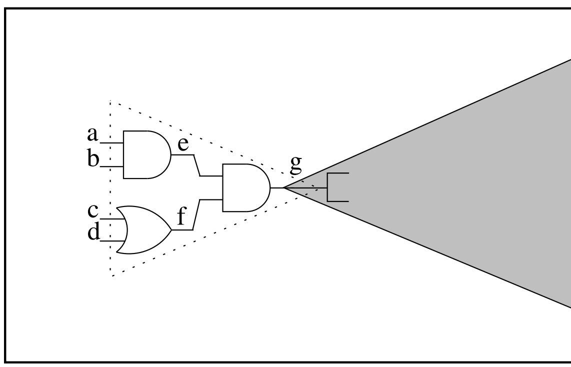

Fig. 3. An example of fanout free region

TABLE II
AN EXAMPLE OF FAULT GROUPING

Fault	Unique signatures
a/1	s_1, s_3
b/1	s_1, s_5
c/0	s_2, s_3
d/0	s_3, s_4
e/1	s_1, s_3, s_5
f/1	s_1, s_2
g/0	s_2, s_3, s_4, s_5
g/1	s_1, s_2, s_3, s_5

through the following example. Consider a fanout free region (FFR) containing three gates of a larger circuit shown in Fig. 3. Assume that for a given set of test patterns, faults in this FFR have the unique signatures for the 8 collapsed stuck-at faults as listed in Table II. A total 21 dictionary entries are required to store the unique signatures for all 8 faults.

For combinational patterns, when a fault within this FFR is detected, for example, a stuck-at 1 ($a/1$) is detected by a pattern P_i as a signature s_3, some downstream faults such as $e/1$ and $g/1$, are guaranteed to be detected by P_i with the same signature s_3. Therefore, many signatures are produced by several different faults. For example, signature s_3 is produced by five faults: $a/1$, $c/0$, $d/0$, $e/1$ and $g/0$. A simple idea to reduce the memory requirement is to group faults with common signatures, and store the shared signatures only once for the fault group, rather than many times seperately for each individual fault. For the example shown in Table II, if we consider all 8 faults as a group, only 5 entries are needed for the 5 unique signatures for this fault group, which is significantly less than 21 entries required by the original method without fault grouping.

In general we group faults in FFRs of the circuit under test and associate the group with unique signatures obtained for any fault in the group. A potential problem for this technique is a larger initial candidate list returned by table lookup during diagnosis, which may increase diagnosis time. For example, for a failing pattern with signature s_5, only 4 faults ($b/1$, $e/1$, $g/0$ and $g/1$) of this FFR are returned as initial candidates by querying the N_{FB} dictionary without fault grouping. However, all 8 faults must be included as initial candidates if fault grouping is used to create the fault dictionary. This potential impact on performance will be examined in the experiments reported in Section IV.

C. The Proposed Diagnosis Algorithm

Next we describe the proposed diagnosis algorithm using the N_{FB} dictionary to speedup diagnosis. First the fault dictionary is computed as previously described. If the fault grouping technique is used, the unique signatures for each fault group, instead for each fault, will be identified and stored. After loading a N_{FB} dictionary, diagnosis for a given fail log proceeds as follows:

For each failing pattern P^i_{fail}, if the number of failing bits is higher than N_{FB}, use the X-algorithm to find the initial candidate list. Otherwise, compute the signature of P^i_{fail} and query the N_{FB} dictionary to find the matching results. If no fault grouping is used, the query results are directly used as initial candidates. Otherwise, the query results (matched fault groups) need to be expanded first, i.e. find all faults belonging to the matched fault groups, and then use the faults as initial candidates. In case no matching result is found by query, the failing pattern is treated as an unexplained failing pattern and may be ignored. After obtaining the initial candidates, the remaining parts of diagnosis process are the same as in a standard effect-cause diagnosis procedure described in Section II.

Since both the X-algorithm and the N_{FB} dictionary querying guarantee to return a list of initial candidates which is a superset of the real suspects, the diagnosis accuracy and resolution are not compromised. The exact same results as the conventional effect-cause diagnosis are guaranteed by the proposed algorithm. In addition, the proposed N_{FB} dictionary efficiently addresses the majority of failing patterns with a small number of failing bits, and thus can achieve a significant performance improvement with minimal memory overhead.

In addition, the proposed N_{FB} dictionary provides a flexible tradeoff between memory and performance. When the memory budget is tight, a small value of N_{FB} can be used to speed up diagnosis with a very small memory overhead. If more memory is available, a bigger dictionary with a higher N_{FB} limit can deliver a better performance speedup.

Unlike [26], where the X-algorithm is completely discarded and initial candidates are identified only from from dictionary query, the proposed algorithm seamlessly integrates X-algorithm and the N_{FB} dictionary, to achieve a significant performance improvement with minimal memory ovehead. It is easy to see that the proposed N_{FB} dictionary without fault grouping will become same as the small dictionary proposed in [26] if N_{FB} is set as $+\infty$.

IV. Experimental Results

In order to validate the proposed techniques, various experiments were performed on the industrial designs described in Table 1 and the results are presented next.

A. Memory Overhead

The memory overhead for the proposed N_{FB} dictionary was first investigated. We preformed experiments on the seven industrial designs listed in Table I. The first experiment is to measure the memory overhead of the N_{FB} dictionary without fault grouping. For each design, the memory usage of the conventional effect-cause diagnosis is used as the baseline. The extra memory used to load the proposed dictionary is considered as memory overhead. The N_{FB} dictionary memory overhead for all seven designs is normalized by dividing the memory usage for loading the design netlist and test patterns and plotted in Fig. 4. Results are reported for values of $N_{FB} = 2, 5, 10, 100, 10^8 (all)$. The dataset $All\text{-}bit$ simply means that all unique signatures with no more than 10^8 failing bits are stored into the N_{FB} fault dictionary, and is actually equivalent to the dictionary of [26].

As we can see, the proposed N_{FB} dictionary can significantly reduce the memory overhead by selecting a reasonable value of N_{FB}, such as 2 or 5, compared to the fault dictionary storing all signatures, and thus is applicable to the largest modern designs. For example, for $D5$, if only signatures for the failing patterns with no more than 2 failing bits are stored, the memory overhead is less than 2% of the total memory needed to load the design and the pattern set. Compared to $All\text{-}bit$, it is about $7X$ smaller. Even if a less aggressive limit $N_{FB} = 5$ is used, the normalized memory overhead, on the average, is still low at 6.4%.

Another experiment was performed to investigate the effectiveness of the proposed fault grouping technique. Similar to the previous experiment, the memory overhead for the N_{FB} dictionary with fault grouping based on fanout-free regions is measured and normalized. The normalized results are plotted in Fig. 5 as $N\text{-}bit + FFR$. For ease of comparison, results for the dictionary without fault grouping are also reported as $N\text{-}bit$. Data for two typical values, $N_{FB} = 2$ and $N_{FB} = 5$, are presented.

It can be seen that the proposed technique of grouping faults based on FFRs, can effectively reduce the memory overhead of the proposed N_{FB} dictionary. For instance, for design $D7$, the fault grouping based on FFRs can reduce the normalized memory overhead of $N_{FB} = 5$ dictionary by about 53.7%, from 9.5% down to 4.4% of the total memory for loading the design netlist and test patterns, which is even smaller than the original $N_{FB} = 2$ dictionary without fault grouping. On average, fault grouping can reduce the memory overhead of the N_{FB} dictionary by 65.0% for $N_{FB} = 2$ and by 48.5% for $N_{FB} = 5$. In addition, the average memory overhead of $N_{FB} = 5$ dictionaries with fault grouping based on FFRs is only 3.3% of the total memory to load design and patterns, and should not be an issue even for very large industrial designs. In cases where the memory budget is extremely tight, we can use the $N_{FB} = 2$ dictionary with fault grouping, whose average memory overhead is less than 1% and thus almost negligible. For easy comparison with the dictionary proposed in [26], the memory overhead of the proposed N_{FB} dictionary is normalized by dividing the memory overhead of $All\text{-}bit$ dictionary, and reported in Table III. As explained before, the $All\text{-}bit$ dictionary is equivalent to the small dictionary proposed in [26]. It can be seen that the memory overhead of the proposed N_{FB} dictionary is significantly smaller than [26]. For example, the $N_{FB} = 2$ dictionary with fault grouping for design $D1$ is only 3.6% of the size of the $All\text{-}bit$ dictionary, i.e. $27.8X$ smaller than [26]. On average, the memory overhead of the $N_{FB} = 2$ dictionary with fault grouping is $17.2X$ smaller than [26]. When N_{FB} is increased to 5, the average size of proposed dictionary with fault grouping is still $5.2X$ smaller than the size of the dictionary in [26]. Even with such a dramatically reduced size, the proposed dictionary can still achieve a similar performance improvement as [26] for most designs.

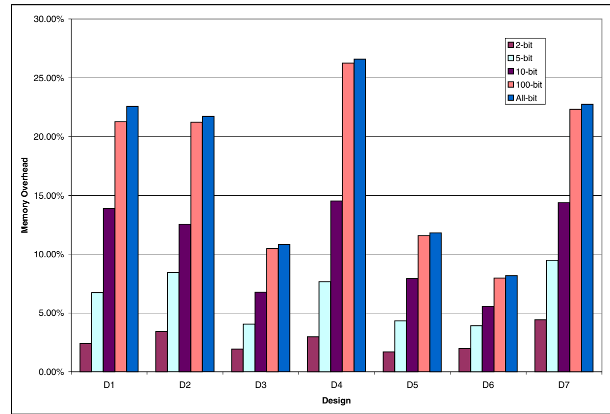

Fig. 4. Memory overhead of N_{FB} dictionary without fault grouping

TABLE III
Memory Overhead N_{FB} vs. Small Dictionary [26]

	D1	D2	D3	D4	D5	D6	D7
5-bit	0.298	0.389	0.375	0.287	0.366	0.479	0.417
2-bit	0.107	0.158	0.178	0.112	0.142	0.243	0.194
5-bit+FFR	0.160	0.202	0.187	0.151	0.216	0.251	0.192
2-bit+FFR	0.036	0.055	0.065	0.038	0.055	0.090	0.065

B. Event Reduction

After investigating the memory overhead of the proposed N_{FB} fault dictionary, experiments were performed to determine performance improvement using the proposed diagnosis

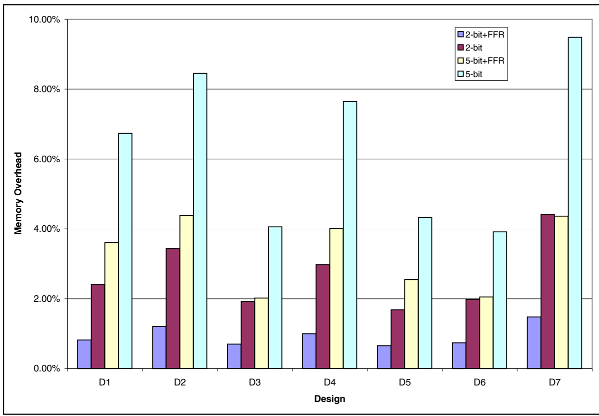

Fig. 5. Memory overhead of N_{FB} dictionary with fault grouping

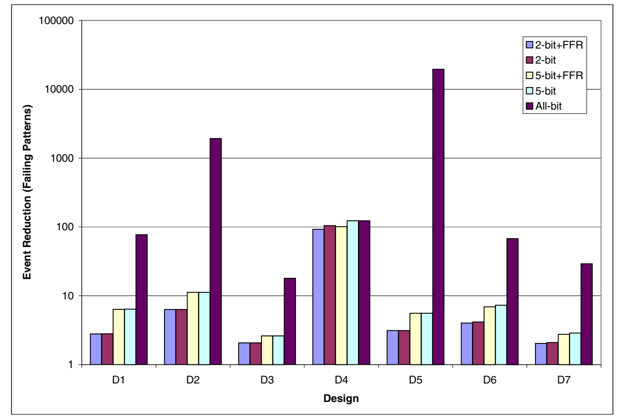

Fig. 6. Reduction of the number of events (failing patterns)

algorithm. For each design listed in Table I, 100 fail logs were created by simulating randomly selected single stuck-at faults. Both clock-related faults and chain-related faults are excluded. A conventional effect-cause diagnosis algorithm is used to diagnose all fail logs, and the results are used as baseline for comparison. The same set of fail logs are also diagnosed by the proposed diagnosis algorithm using the N_{FB} fault dictionary with/without utilizing fault grouping technique, $N_{FB} = 2, 5$. Since only the failing pattern processing is targeted in this work, our analysis will focus on the improvement for processing failing patterns. For different diagnosis runs, the number of events triggered and the CPU time consumed during processing failing patterns are used as metrics for evaluating performance. An event is defined as evaluating the value of a gate during fault simulation. Unlike CPU time, the number of events is independent of run time environment and specific implementation of the procedures.

We first investigated the reduction of the number of events triggered during simulating failing patterns on the initial set of candidates by the proposed algorithm with the N_{FB} fault dictionary. For each design, the total number of events is computed for 100 fail logs diagnosed. The reduction of the number of events is computed by dividing the total number of events for the conventional effect-cause diagnosis by the number of the proposed algorithm. Results are plotted in Fig. 6. It can be seen that the total number of events triggered during simulating failing patterns can be dramatically reduced by using the proposed algorithm, and thus the diagnosis performance can be significantly improved. For design $D4$, the N_{FB} dictionary can help to reduce the number of events by about $100X$. On average, $16.1X \sim 22.7X$ reductions in the number of events are achieved by four different dictionaries.

Another interesting observation is that the proposed fault grouping technique based on FFRs causes a minimal increase in the number of events relative to the case when fault grouping is not used. For example, for the $N_{FB} = 2$ dictionaries, the average reduction in the number of events only drop by 9.6%, from $17.8X$ down to $16.1X$, when fault grouping based on FFRs is adopted. The reason is that the extra candidates introduced by fault grouping typically have a local effect and can be quickly dropped during simulation. Therefore, there is no significant impact on the reduction of the number of events. Because of the significant saving in dictionary size using fault grouping, for example, on average 65.0% saving for $N_{FB} = 2$, fault grouping should be preferred.

C. Run Time Speedup

The speedup of run time for processing failing patterns was also investigated. The average CPU time for analyzing all failing patterns for a fail log is computed for all 100 fail logs for each design. The speedup for each run is computed by dividing the average CPU time of the standard effect-cause algorithm by the run time of the proposed procedure. The same set of N_{FB} dictionaries as above are used and the results are presented in Fig. 7.

As we can see, a significant performance improvement is achieved by the proposed algorithm using $N_{FB} = 2, 5$ dictionaries. On average, a $2.9X \sim 4.3X$ speedup is achieved by different runs for the seven designs. For design $D6$, higher than $6.0X$ speedup is achieved by the proposed algorithm with $N_{FB} = 5$ dictionaries, both with and without fault grouping. In addition, it can be seen that the $N_{FB} = 5$ dictionary with fault grouping has a very similar speedup as the dictionary without fault grouping, but a much smaller memory overhead, which makes it a preferred tradeoff.

Compared to the extreme case, where $N_{FB} = 10^8$, the $N_{FB} = 5$ dictionary with fault grouping can achieve a similar high speedup for four out of seven designs, $D1$ to $D4$, with a much smaller memory overhead. For the other three designs, $D5$ to $D7$, there is still plenty of room for improving performance by increasing N_{FB}, which means a

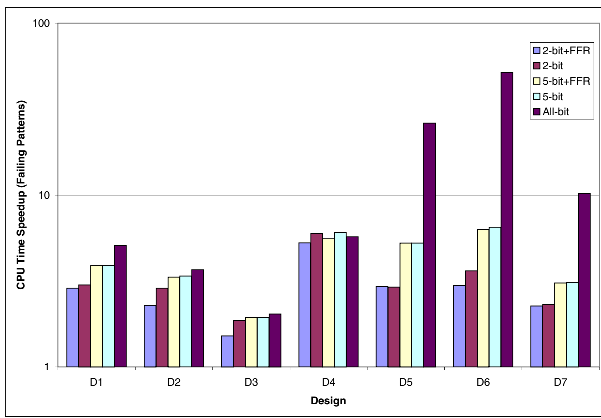

Fig. 7. CPU time speedup (failing patterns)

higher speedup can be achieved by using a larger dictionary if more memory is available. For example, the speedup for $D5$ can be increased from $5.3X$ of the $N_{FB} = 5$ dictionary with fault grouping to $26.2X$ using *All-bit* with a cost of $4.6X$ memory overhead. One proper way to use the proposed techniques is to first determine the memory budget and then create a N_{FB} dictionary as large as can be accommodated to achieve a higher performance.

V. Conclusions

We proposed a new N_{FB} dictionary with minimal size which seamlessly works with the X-algorithm normally used in effect-cause diagnosis procedures to speed up the conventional effect-cause algorithm. The diagnosis performance is significantly improved by replacing X-algorithm with efficient dictionary query for failing patterns with a small number of failing bits, which drastically reduces the number of events triggered by simulating failing patterns. The same high accuracy and resolution of diagnosis results as the standard effect-cause diagnosis is guaranteed. The memory overhead of the proposed N_{FB} dictionary is minimized by only storing unique signatures for failing patterns with no more than N_{FB} failing bits, and grouping faults based on fanout-free regions. The minimal memory overhead makes sure that the proposed dictionary and the diagnosis algorithm is applicable to the modern large designs with tens of millions of gates. Flexible tradeoffs between memory overhead and performance improvment can be easily achieved through the configurable parameter N_{FB} of the proposed dictionary.
As to the creation time for the proposed dictionary, it only takes a few hours for the largest design $D5$. Therefore such one-time cost could be justifiable when one needs to process thousands of failed dies.

References

[1] M. Abramovici and M. A. Breuer, *Fault diagnosis based on effect-cause analysis: an introduction*, Proc. of DAC, 1980, pp. 69-75

[2] S. B. Akers, S. Park, B. Krishnamurthy, and A. Krishnamurthy, *Why is less information from logic simulation more useful in fault simulation*, Proc. of ITC, pp. 786-800, 1990

[3] M. E. Amyeen, D. Nayak, and S. Venkataraman, *Improving precision using mixed-level fault diagnosis*, Proc. of ITC, paper 22.3, 2006

[4] D. Appello, A. Fudoli, K. Giarda, E. Gizdarski, B. Mathew, and V. Tancorre, *Yield analysis of logic circuits*, Proc. of VTS, pp. 103-108, 2004

[5] T. Bartenstein, D. Heaberlin, L. Huisman, and D. Slinwinski, *Diagnosing combinational logic designs using the single location-at-a-time (SLAT) paradigm*, Proc. of ITC, 2001, pp. 287-296

[6] T. Bartenstein, B. Koenemann, L. Todd, and L. Valerie, *Integrating logical and physical analysis capabilities for diagnositics*, Proc. of ISTFA, pp. 521-526, 2004

[7] D. Bodoh, A. Blakely, and T. Garyet, *Diagnostic fault simulation for the failure analyst*, Proc. of ISTFA, pp. 181-190, 2004

[8] V. Boppana, I. Hartantet, and W. K. Fuchs, *Full fault dictionary storage based on labelled tree encoding*, Proc. of VTS, 1996, pp. 174-179,

[9] W.-T. Cheng, K.-H. Tsai, Y. Huang, N. Tamarapalli, and J. Rajski, *Compactor independent direct diagnosis*, Proc. of ATS, pp. 204-209, 2004

[10] B. Chess and T. Larrabee, *Creating small fault dictionaries*, IEEE TCAD, Vol. 18, No. 3, 1999, pp. 346-356

[11] H. Erb, C. Burmer, and A. Leininger, *Yield enhancement through fast statistical scan test analysis for digital logic*, Proc. of Adv. Semi. Manuf. Conf. and Workshop, pp. 250-255, 2005

[12] C. Hora, R. Segers, S. Eichenberger, and M. Lousberg, *An effective diagnosis method to support yield improvment*, Proc. of ITC, pp. 260-269, 2002

[13] M. Keim, N. Tamarapalli, H. Tang, M. Sharma, J. Rajski, C. Schuermyer, and B. Benware, *A rapid yield learning flow based on production integrated layout-aware diagnosis*, Proc. of ITC, paper 7.1, 2006

[14] B. Kruseman, A. Majhi, C. Hora, S. Eichenberger, and J. Meirlevede, *Systematic defects in deep sub-micron techologies*, Proc. of ITC, pp. 290-299, 2004

[15] D. B. Lavo, I. Hartanto, and T. Larrabee, *Multiplets, models, and the search for meaning: improving per-test fault diagnosis*, Proc. of ITC, 2002, pp. 250-259

[16] A. Leininger, P. Muhmentaler, W.-T. Cheng, N. Tamarapalli, W. Yang, and H. Tsai, *Compression mode diagnosis enables high volume monitoring diagnosis flow*, Proc. of ITC, paper 7.3, 2005

[17] I. Pomeranz and S. M. Reddy, *On the generation of small dictionaries for fault location*, Proc. of ICCAD, pp. 272-279, 1992

[18] I. Pomeranz, S. Venkataraman, S. M. Reddy, and B. Seshadri, *Z-sets and z-detections: circuit characteristics that simplify fault diagnosis*, Proc. DATE, 2004, pp. 68-73

[19] C. Schuermyer, K. Cota, R. Madge, and B. Benware, *Identification of systematic yield limiters in complex ASICS through volume structural test fail data visualization and analysis*, Proc. of ITC, paper 7.1, 2005

[20] H. Tang, S. Manish, J. Rajski, M. Keim, and B. Benware, *Analyzing volume diagnosis results with statistical learning for yield improvement*, Proc. of ETS, 2007, pp. 145-150

[21] S. Venkatarman and S. Drummonds, *POIROT: a logic fault diagnosis tool and its application*, Proc. of ITC, 2000, pp. 253-262

[22] J. A. Waicukauski and E. Lindbloom, *Failure diagnosis of structured circuits*, IEEE Design and Test of Computers, Vol.6, No. 4, 1989, pp. 49-60

[23] Z. Wang, M. M. Sadowska, K.-H. Tsai, and J. Rajski, *Analysis and methodology for multiple-fault diagnosis*, IEEE TCAD, Vol. 25, No. 3, 2006, pp. 558-575

[24] W. Zou, W.-T. Cheng, and S. M. Reddy, *Bridge defect diagnosis with physical information*, Proc. of ATS, pp. 248-253, 2005

[25] W. Zou, W.-T. Cheng, S. M. Reddy, and H. Tang, *On methods to improve location based logic diagnosis*, Proc. of VLSI Design, 2006, pp. 181-187

[26] W. Zou, W.-T. Cheng, S. M. Reddy, and H. Tang, *Speeding up effect-cause defect diagnosis using a small dictionary*, Proc. of VTS, paper 6B-2, 2007