

Line Oriented Structural Equivalence Fault Collapsing

Mehran Nadjarbashi, Zinalabedin Navabi and Mohammad R. Movahedin
nadj@cad.ece.ut.ac.ir, navabi@ece.neu.edu, mov@usa.net
Electrical and Computer Engineering Department
Faculty of Engineering – Campus #2 – University of Tehran
14399, Tehran – IRAN

Abstract

This paper presents a new perspective on structural fault collapsing at the gate level. As compared with current structural fault collapsing methods, a method based on this point of view enables a faster and optionally a more reduced fault collapsing. In addition, our new fault collapsing method is easily implementable in the VHDL language. Performance improvement of this method has been verified using ISCAS benchmarks.

1. Introduction

In a digital system test process, fault collapsing is the process of reducing number of faults to only those which can be distinguished. Since test generation algorithms are time-consuming, it is important that a fault list used in a test process (Fig.1) is reduced as much as possible. A small fault list reduces redundancy in generated test vectors as well as the overall test time. In this paper, we consider combinational logic circuits and use the single stuck-at-fault model. In addition, we assume that faults in the original fault list are detectable. Because of advantages of equivalence fault collapsing [1], i.e., ability to locate site of faults, we have only considered equivalence reduction in our method and implementation.

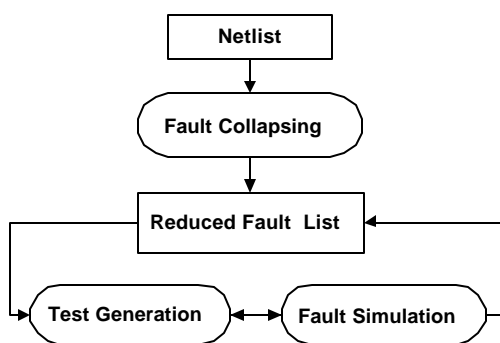


Figure 1. A general test CAD flow

Two different faults are functionally equivalent if occurrence of either fault results in circuits that are functionally identical [2]. In other words, no test vector

can be found to distinguish between two faulty circuits. For example, there is no way to distinguish between the presence of stuck-at 0 on any input and stuck-at 0 on the output of an AND gate.

Section 2 describes structural fault collapsing and discusses how it is different from functional. Section 3 discusses present fault collapsing methods. These methods are generally gate oriented, and we will suggest improvements to these methods. In section 4, a new strategy of fault collapsing, which is mainly based on faults on circuit lines will be presented. Section 5 improves this method, and section 6 is on results and comparisons.

2. Functional and Structural Fault Collapsing

Functional fault collapsing relations cannot be directly applied, because it is hard to exhaustively select two different faults from a fault list and check their equivalency. Furthermore, determining whether two arbitrary faults are functionally equivalent is an NP-complete problem [3]. For example, determining that in Fig. 2, $d\text{-}sa1$ and $j\text{-}sa1$ are functionally equivalent is not a simple task.

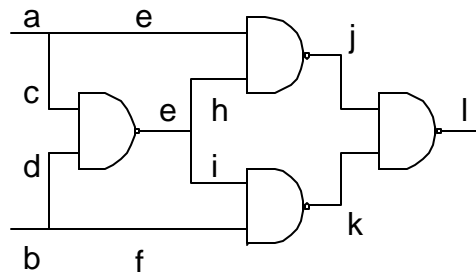


Figure 2. $d\text{-}sa1$ and $j\text{-}sa1$ are functionally equivalent

Instead of doing a functional fault collapsing (FFC), we usually use structural fault collapsing (SFC), in which structural relations dominate the fault collapsing process. In this technique, two faults are said to be structurally equivalent if the effect of applying each of these faults alone to the circuit creates circuits that are structurally identical. For example, in Fig. 2,

two faults $h-sa0$ and $j-sa1$ are structurally equivalent. This is because if we apply each of these faults separately and remove the lines on which faults cause constant values then the corresponding simplified circuits will be identical.

3. Implementation Methods of SFC

The classic method of SFC is presented by McCluskey and Clegg in [2], and is based on two graph-transform algorithms that produce the related functional equivalence classes. From each class, only one fault as the representative fault will be selected and all other members of that class will be collapsed into this representative fault. Based on the fundamental concepts, several implementation methods for SFC are presently available. The simplest method is called "Local Equivalent Fault Collapsing", which is based on the fact that for a logic gate with n -inputs and one output, there may be $2(n+1)$ possible single stuck-at faults. For an AND gate, stuck-at 0 faults at all inputs and the output are functionally equivalent. Thus, distinguishable faults in an n -input gate are limited to $n+2$. Applying this technique to a complete circuit is shown in the example of Fig. 3. In this circuit, initially all possible faults on gate inputs and outputs are identified (solid dots represent SA1 faults). Then local collapsing rules as discussed for an AND gate are applied to reduce circuit faults as shown in Fig. 3.

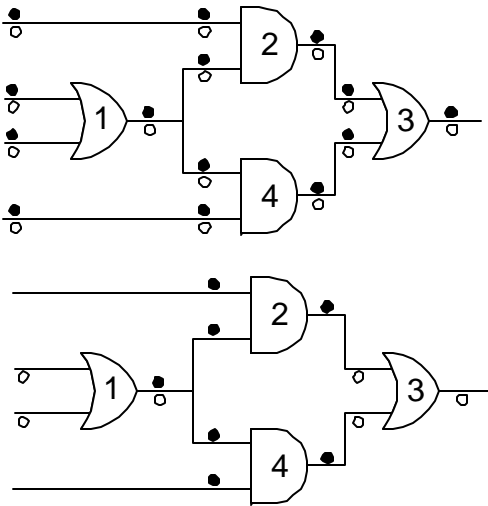


Figure 3. Local equivalent fault collapsing

Note that faults on either end of a line are the same, also faults on the stem and branches of a fanout are not functionally equivalent.

Modifying this technique by adding the following rules improves collapsing:

1) Always put a fault that is representative of several input faults on the output of the gate that is subject of local collapsing.

2) Apply Rule 1 to the gates at the lower structural levels first.

The level of a gate in a circuit is defined as its position with respect to circuit primary inputs, the longest path to primary inputs is considered. Therefore, a gate with an input connected to another gate, is at least one level higher than the gate providing its input. In order to form a data structure for the implementation of this rule, we must sort our circuit representation before fault collapsing takes place. Applying the above rules to the circuit of Fig. 3, results in reduction of faults by one, as shown in Fig. 4.

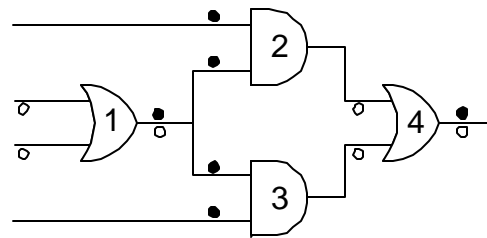


Figure 4. Result of modifying gate oriented collapsing

To apply these rules to single-input gates, e.g., an inverter, both input stuck-at faults are moved to the output of the gate.

This process and similar approaches of structural equivalence fault collapsing reduce the initial set of faults by about 50 percent [1].

4. Line oriented SFC

A fast implementation of fault collapsing can be achieved when we view faults on lines instead of those on gate ports. Our approach is based on the classic concepts, and is only different from previous implementations in its view of placing faults. Instead of initially arranging all possible faults on the ports of gates and then trying to collapse them, we only place those faults on circuit lines that cannot collapse any further. The criterion used in placing a specific stuck-at fault on a line is based on the gate driven by the line. Table 1 shows faults to be placed on lines, based on the gate the line is driving.

As with most test applications, fanout is treated as an actual component.

An example is shown in Fig 5. Faults shown on circuit lines are as determined by entries of Table 1.

Table 1

Type of target gate	Put this (these) faults on the line
AND, NAND	SA1
OR, NOR	SA0
INV, BUF	None
FANOUT	SA0, SA1
XOR	SA0, SA1
Primary Output	SA0, SA1

This approach does not have the overhead of previous methods. We have eliminated the need for sorting gates or signals in a data structure for this process. Furthermore, each line of circuit is processed only once in any given order.

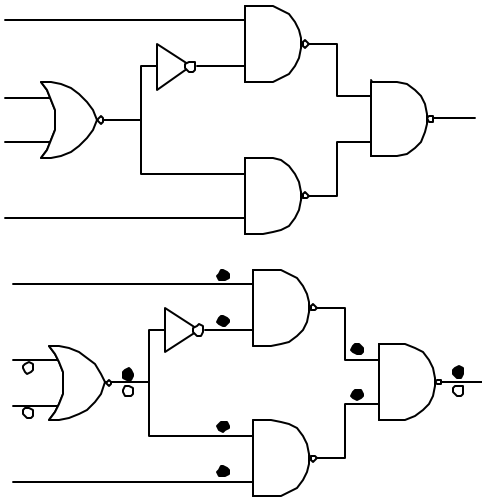


Figure 5. Placing suitable faults on circuit lines

Not requiring sorting of circuit information eliminates the need for large memory usage. This becomes important when processing circuits with tens of thousands of gates.

Another advantage of this method is that it can easily be implemented using the VHDL programming environment [4]. Because our method concentrates on line faults and, unlike gates, all lines in a circuit follow the same model, implementation of fault collapsing in VHDL only requires a single VHDL line architecture. A previous work in this area used separate VHDL models for all types of logic gates [5].

5. Maximal SFC

A special case occurs in stems of reconvergent fanouts. Such faults may be collapsed into gate outputs at the reconvergence point. Consider, for example, circuit in Fig. 4. The stuck-at 0 fault on the fanout stem in this circuit is structurally equivalent to the stuck-at 0

fault on the output line. This equivalence is not detected by any of the existing collapsing methods. With an extension of the method presented in Section 4, we will be able to find faults at fanout stems that collapse into gate output lines at the reconvergence nodes.

We can think of these faults as bubbles at the fanout stem. Such a bubble propagates through all branches of the fanout, that go through gates until they reach at the reconvergence point. While propagating, a stuck-at 0 bubble can only pass through AND and NAND gates, while a stuck-at 1 bubble can only pass through OR and NOR gates. If a stuck-at 0 bubble pass through an invert function (NAND, NOR, INV gates), it will be transformed to a stuck-at 1 bubble, and vice versa. Considering this propagation of bubbles, if all bubbles reach inputs of a gate at the reconvergence node, the fault at stem can be removed.

Although this method detects more equivalent faults in the reconvergent parts, the number of such faults in circuits we tested is on the average about 2% of the original faults.

6. Implementation

We implemented our fast, line oriented SFC using LTEST v1.0 package [6], and compared performance of our method with that of another method that is based on gate faults that its data structure is already sorted. The existing fault collapsing in LTEST package is based on gate faults, but it is very slow compared to our own gate oriented implementation.

For several of the ISCAS-85 circuits, run time gain of line oriented method are shown in Table 2. Obviously, this gain will increase if we need to sort our data structure in a gate-oriented implementation. Collapsing in Pass 1 is the result of application of the method presented in Section 4, and Pass 2 is after application of the bubble method of Section 5. In most cases, reduction in number of faults in Pass 2 is not justified by the run time required for this collapsing.

7. Conclusions

This paper presented a synopsis on fault collapsing and emphasizing on structural fault collapsing methods at the gate level. We presented a method that is based on line faults instead of gate faults. Compare to existing SFC methods, our method has several advantages including a better run-time. Performance improvements were verified in this paper. We have also presented a collapsing method for faults that correspond to reconvergent fanouts, which are generally classified as hard-to-detect faults. An interesting observation is that

a better run-time improvement is obtained for larger circuits. This, in addition to the low memory requirement of our method, makes this technique a

useful one for fault collapsing in large gate level circuits.

Table 2. The achieved results using ISCAS-85 benchmarks

Benchmark	Number of gates	Original faults	Percent of faults collapsed in pass 1	Percent of faults collapsed in pass 2	Speed-gain achieved in line oriented method
c499	514	2248	47.7	0	21.5%
c880	383	1780	47.3	1.3	31.8%
c1355	546	2728	45.5	3.8	33.9%
c1908	880	3542	46.2	0.5	36.1%
c2670	1193	4892	49.4	3.1	38.1%
c3540	1669	6804	49.3	1.9	41.5%
c5315	2307	10228	48.1	2.2	44.6%
c6228	2416	12704	42.1	4.1	46.9%
c7552	3512	14216	47.4	1.2	48.4%
Average			47.0	2.0	38.1%

References

- [1] M. Abramovici, M.A. Breuer and A.D. Friedman, *“Digital Systems Testing and Testable Design”*, IEEE Press, New Jersey, 1990.
- [2] E.J. McCluskey and F.W. Clegg, *“Fault Equivalence in Combinational Logic Networks”*, IEEE Trans. Computers, Vol C-20, pp. 1286-1293, November 1997.
- [3] A. Goundan, *“Fault Equivalence in Logic Networks”*, Ph.D. Thesis, University of Southern California, 1978.
- [4] Z. Navabi, *“VHDL: Analysis and Modeling of Digital Systems”*, McGraw Hill, New York, 1993.
- [5] Z. Navabi and M. Shadfar, *“A VHDL Based Test Environment Including Models for Equivalence Fault Collapsing”*, Proceedings of VHDL International Users' Forum. May 1-4, 1994, Oakland, CA.
- [6] I. Parulkar and M. Lempel, *LTEST v1.0*, Test Software Package, University of Southern California, 1991.