

EFFICIENT GLOBAL FAULT COLLAPSING FOR COMBINATIONAL LIBRARY MODULES

Hussain Al-Asaad
Department of Electrical & Computer Engineering
University of California
Davis, CA, U.S.A.

Abstract—*Fault collapsing is the process of reducing the number of faults by using redundancy and equivalence/dominance relationships among faults. Exact global fault collapsing can be easily applied locally at the logic gates, however, it is often ignored for library modules due to its high demand of resources such as execution time and/or memory. In this paper, we present an efficient and exact global fault collapsing method for library modules that uses both binary decision diagrams and fault simulation with random vectors. Experimental results show that the new method reduce the number of faults drastically with feasible resources and produce significantly better results than existing approaches.*

Keywords: Global fault collapsing, fault simulation, testing, combinational circuits.

1 INTRODUCTION

To test a digital circuit, an automatic test pattern generation (ATPG) tool generates a test set that targets possible physical faults. As the complexity of the digital circuit increases, the possible number of physical faults increases that consequently leads to a significant slow down of the test generation process using the ATPG tool. One approach for considerably reducing the length of the testing process as well as producing compact test sets is fault collapsing. Fault collapsing [1] is the process of reducing the number of faults by using redundancy, equivalence, and dominance relationships among faults. Exact fault collapsing can be easily applied locally at the logic gates; however, it is often not feasible to apply it globally for large circuits.

Several researchers have worked on fault collapsing. An algorithm was presented in [2] that collapse all the structurally equivalent faults in a circuit, plus many of the functionally equivalent faults. Application of the algorithm to the ISCAS-85 benchmark circuits [3] establishes that identification of functionally equivalent faults is feasible, and in some cases, they are a large fraction of the faults in a circuit. However, the overall produced collapsed fault list is still large in comparison to the global collapsed fault list.

A graph-theoretic hierarchical fault collapsing method was presented in [4][5] that can collapse faults in any large cell-based circuit. Since functional analysis (equivalence and dominance) is computationally expensive, it is only applied to standard cells. As an example, consider the size of the collapsed fault list for

an exclusive-OR cell. Using the method of [5], the collapsed fault list reduces to just four faults when functional fault collapsing is considered. With the traditional method of structural collapsing this set contains 13 faults. When the exclusive-OR cell is used to build an 8-bit adder circuit, the size of the collapsed fault list produced by [5] reduces to 112 faults from a total of 466 faults. Traditional structural fault collapsing would have given a set of 226 faults. Although a significant reduction is achieved here, the method assumes a hierarchical design with a good use of standard cells. Moreover, the size of the produced collapsed fault list is still large in comparison to the exact global collapsed fault list.

Recently, a new diagnostic and detection fault collapsing method was introduced for multiple-output circuits [12]. Using this method, a significant reduction in the fault list was achieved, however, the method again assumes a hierarchical design (such as adders and ALUs) with a good use of small standard cells.

Efficient techniques for identifying functionally equivalent faults in combinational circuits were presented in [6]. The techniques are based on implication of faulty values, and evaluation of faulty functions in cones of dominator gates of fault pairs. Experimental results show that most of the equivalent fault pairs are identified. However, this work does not aim at producing a small collapsed fault list.

In our previous work [7], we have presented a preliminary method that produces a compact fault list—an approximation of the global collapsed fault list. Our approximate global fault collapsing technique is based on the simulation of random vectors. Experimental results show that our method produced significant reduction in the size of the collapsed fault list. However, our preliminary approximate global fault collapsing tool (a set of scripts manipulating several academic CAD tools) turned out to be resource intensive and memory hungry process. Even with only 1,000 test vectors, many of the smaller benchmark circuits required several hours to simulate.

Recently, we have further introduced two new methods for global fault collapsing: an exact method (EGFC-BDD) using binary decision diagrams [8], and an approximate method (AGFC) using fault simulation with random vectors [9]. We describe these methods in Section 2. In Section 3, we present our new efficient method of exact global fault collapsing (EGFC-HYB) that uses both binary decision diagrams and fault simulation with random vectors. The method integrates EGFC-BDD and AGFC to take the advantages of both

and consequently produce fast exact global fault collapsing. The section also presents some experimental results. Finally, we conclude the paper in Section 4 and present some comparisons and possible extensions.

2 FAULT COLLAPSING

In physical fault testing, physical defects are abstracted into a logical fault model. The most widely-used logical fault model is the Single Stuck-Line (SSL) model [1]. Under this model, every single signal line can become permanently fixed (stuck) at a logical 1 or 0 value. The model is simple and technology-independent. It represents a large number of different physical faults, and tests derived for SSL faults detect many design errors/faults. In this paper, we only consider SSL faults; however, our method is applicable to several other fault models.

Fault collapsing first removes redundant faults from the fault list. A fault is redundant if there is no test that can detect it. In other words, a fault is redundant if the faulty function is the same as the correct function. Fault collapsing then reduces the number of faults using two relationships among faults: fault equivalence and fault dominance. Two faults are considered equivalent if the faulty functions (for the case of a single-output circuit) produced by the two faults are equal. Alternatively, the two faults are equivalent if they can be detected by the same tests. In this case, there is no way to distinguish between the two faults. For example, the SSL fault a stuck-at 0 represented by $a/0$ in Figure 1 is equivalent to the fault $z/0$. If two faults are equivalent then one of the faults can be dropped from the fault list since the detection of the other fault guarantees the detection of the dropped fault.

A fault f is considered to dominate another fault g when every test for g is also a test for f . For example, the fault $z/1$ dominates the fault $a/1$ in Figure 1 since the only test vector 01 for $a/1$ (shaded in the figure) is also a test for $z/1$. If a fault f dominates a fault g , then the fault f can be dropped from the fault list since the detection of g guarantees the detection of f .

By applying fault collapsing to the AND gate in Figure 1, we can reduce the number of faults from six to three. First, there are no redundant faults on the AND gate that should be dropped. Second, the faults $a/0$ and $b/0$ are dropped since they are equivalent to $z/0$. Finally, the fault $z/1$ is dropped since it dominates both $a/1$ and $b/1$. The collapsed fault list is thus $\{a/1, b/1, z/0\}$. A test set that detects the faults in the collapsed list can be derived from the table in Figure 1 as $\{01, 10, 11\}$. This test detects all faults in the collapsed fault list and con-

Inputs		Correct function (z)	Faulty functions					
a	b		a/0	a/1	b/0	b/1	z/0	z/1
0	0	0	0	0	0	0	0	1
0	1	0	0	1	0	0	0	1
1	0	0	0	0	0	1	0	1
1	1	1	0	1	0	1	0	1

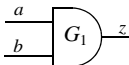
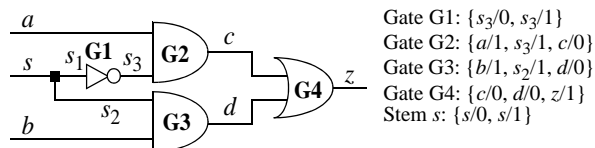


Figure 1 Fault collapsing for a 2-input AND gate.



Inputs			Correct output z	Faulty functions									
s	a	b		s/0	s/1	s ₃ /0	s ₃ /1	a/1	b/1	s ₂ /1	c/0	d/0	z/1
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	1	0	0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	0	0	1	1	1	1	0	1	1
0	1	1	1	1	1	0	1	1	1	1	0	1	1
1	0	0	0	0	0	0	0	0	1	0	0	0	1
1	0	1	1	0	1	1	1	1	1	1	1	0	1
1	1	0	0	1	0	0	1	0	1	0	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1	0	1

Figure 2 A simple multiplexer circuit with a list of its gate faults and the resulting fault table.

sequently all six faults in the AND gate.

There are two approaches to fault collapsing: local and global. The local fault collapsing method computes the collapsed fault list for individual gates and then collects the collapsed fault lists for the gates to form the overall collapsed fault list for circuit. For example, by using fault collapsing over the gates in the circuit shown in Figure 2, we get the results shown in the figure. Both stuck at faults on line s (called a stem since it branches to other lines) need to be considered because s is not an input or output of any gate. Using local fault collapsing, we combine the faults on the gates to form the collapsed fault list for the circuit as $\{s/0, s/1, s_3/0, s_3/1, a/1, b/1, s_2/1, c/0, d/0, z/1\}$. Therefore, by using local fault collapsing we were able to reduce the fault list from 18 to 10.

Global fault collapsing is similar to local fault collapsing, except that we perform the same process of fault collapsing on the entire circuit as opposed to individual gates. In other words, we look for equivalent and dominance relationships among all faults in the circuit. For example, to perform global fault collapsing for the circuit in Figure 2, we compute a table for all faulty functions (called a fault table) as shown in Figure 2. It is simpler to start with the local collapsed fault list since it has less faults than the original fault list for the circuit. We then drop faults from the local collapsed fault list using redundancy, equivalence and dominance relationships. It is obvious that there are no redundant faults that should be dropped. The faults $s/0, b/1, z/1$ are dropped since they dominate $s_3/1$. Also, the faults $s/1, a/1$ can be dropped since they dominate $s_2/1$. The fault $s_3/0$ is dropped since it is equivalent to $c/0$. The global collapsed fault list for the circuit is thus $\{s_2/1, s_3/1, c/0, d/0\}$. Hence, by using global fault collapsing we were able to reduce the number of faults from 18 to 4. This is in effect a 77.78% reduction from the original fault list.

Local fault collapsing can be easily scaled to large circuits. However, global fault collapsing is often

avoided due to the lack of resources including the expensive computations and memory needed to determine redundancy, equivalence and dominance relationships among the faults in the overall circuit. In the rest of this section, we describe our recent new techniques for global fault collapsing: EGFC-BDD and AGFC.

2.1 BDD-BASED EXACT GLOBAL FAULT COLLAPSING (EGFC-BDD)

In this sub-section, we first determine the conditions needed to establish redundancy, equivalence, and dominance. For this purpose, let us consider a combinational circuit C with n inputs, m outputs, and k faults. Let the n inputs of C be represented as $X = x_{n-1} \dots x_1 x_0$. Also, let the m outputs of C be represented as $Y = y_{m-1} \dots y_1 y_0$. For every fault f of C , we define the function T as follows:

$$T(X, f) = \sum_{i=0}^{m-1} (y_i(X, c) \oplus y_i(X, f))$$

where $y_i(X, c)$ is the correct (fault-free) function of output i and $y_i(X, f)$ is the faulty function of output i in the presence of f . In fact, the function T specifies whether an input X is a test for f or not as follows:

$$T(X, f) = 1 \text{ if } X \text{ is a test for } f$$

$$T(X, f) = 0 \text{ if } X \text{ is not a test for } f$$

As a consequence of the above, a fault f is redundant if and only if $T(X, f) \equiv 0$. This translates to:

$$T(X, f) = \sum_{i=0}^{m-1} (y_i(X, c) \oplus y_i(X, f)) \equiv 0$$

which ultimately leads to $y_i(X, f) \equiv y_i(X, c)$ for every i . So, to prove that a fault f is redundant, we need to compute the faulty functions of the outputs and show that they match the correct (fault-free) functions of the corresponding outputs.

The equivalence relationship between two faults f and g is defined as: $T(X, f) \equiv T(X, g)$. For the case of a single-output circuit ($m = 1$), the above equation reduces to $y_0(X, f) = y_0(X, g)$. So, the two faults f and g are equivalent if they have the same faulty function (as discussed earlier in Section 2). For the case of multi-output circuit, if the faulty functions are equivalent for every output, then the two faults are equivalent. However, the converse is not necessarily true.

The dominance relationship between a fault f and a fault g is defined as follows:

$$f \text{ dominates } g \Leftrightarrow \overline{T(X, f)} T(X, g) \equiv 0$$

In other words, there is no test for g that is not a test for f . So, to prove that f dominates g , we need to compute the function $D_{fg}(X, f, g) = \overline{T(X, f)} T(X, g)$ and prove that it is identical to zero. Similarly, to prove that g dominates f , we need to compute the function $D_{gf}(X, f, g) = T(X, f) \overline{T(X, g)}$ and prove that it is identical to zero.

It can be easily shown that if f dominates g and g dominates f , then f is equivalent to g . The proof is as follows:

$$f \text{ dominates } g \Leftrightarrow \overline{T(X, f)} T(X, g) \equiv 0$$

$$g \text{ dominates } f \Leftrightarrow T(X, f) \overline{T(X, g)} \equiv 0$$

$$(T(X, f) \overline{T(X, g)} + \overline{T(X, f)} T(X, g)) \equiv 0$$

$$T(X, f) \oplus T(X, g) \equiv 0$$

$$T(X, f) \equiv T(X, g)$$

Hence, f is equivalent to g . Based on the above, we need to compute the two functions $D_{fg}(X, f, g)$ and $D_{gf}(X, f, g)$ to determine the relationship between f and g as follows:

- $D_{fg}(X, f, g) \equiv 0$ & $D_{gf}(X, f, g) \equiv 0$: f is equivalent to g .
- $D_{fg}(X, f, g) \equiv 0$ & $D_{gf}(X, f, g) \neq 0$: f dominates g .
- $D_{fg}(X, f, g) \neq 0$ & $D_{gf}(X, f, g) \equiv 0$: g dominates f .
- $D_{fg}(X, f, g) \neq 0$ & $D_{gf}(X, f, g) \neq 0$: f is not related to g .

Once the relationship between f and g is established, we can possibly drop a fault as follows:

- If f is equivalent to g then drop f or g .
- If f dominates g then drop f .
- If g dominates f then drop g .
- If f is not related to g then no fault is dropped.

In order to produce the collapsed fault list, we need to compute the functions $T(X, f)$, $D_{fg}(X, f, g)$, and $D_{gf}(X, f, g)$. We use reduced ordered binary decision diagrams (ROBDDs) [10] in the computations of functions. This is the case since ROBDDs is a compact canonical representation that can be easily manipulated. Moreover, algorithms for ROBDD operations are well studied and are widely used in various research fields including test, synthesis, and verification.

In using ROBDDs, there is a trade-off between the computation time of functions and the memory needed for the computations. Our EGFC-BDD tool stores all ROBDDs for internal signals so that it can compute the functions $T(X, f)$, $D_{fg}(X, f, g)$, and $D_{gf}(X, f, g)$ as quickly as possible. EGFC-BDD can be easily modified so that it uses less memory on the expense of more execution time.

Our EGFC-BDD tool implements the method presented in this section for the elimination of faults from the fault list. It is written using C++ in approximately 7000 lines of code. A detailed description of the EGFC-BDD algorithm as well as experimental results that illustrate its capabilities can be found in [8].

2.2 SIMULATION-BASED APPROXIMATE GLOBAL FAULT COLLAPSING (AGFC)

In our AGFC method, a large set of random vectors is used to reduce the number of faults instead of using the complete vector set for the library module. The idea behind approximate collapsing is that the resulting faults after the simulation is an approximation of the faults from exact global fault collapsing of the library module. As more and more vectors are applied for the simulation, the results appear more and more similar to those of exact global fault collapsing.

In order to identify redundant faults, our approximate global fault collapsing method works as follows.

We first label all the faults in the circuit as redundant (r) faults since we have no information about the detectability of the faults. We then apply a random test vector t and determine the faults detected by t and then update the type of every fault. Figure 3 shows a state diagram for tracking the types of all faults in the circuit. Note that all faults are initially in the redundant (r) state and once a fault becomes detectable (d) due to detection by a test t , it remains in that state forever.

To speed up the fault collapsing in our implementation, we apply a packet of 32 random vectors and determine the faults detected by the packet and then update the type of every fault. The type of a fault changes to detectable (d) if the fault is detected by at least one vector from the packet. The process is repeated for several iterations until no change is reported in the types of faults for a constant number of random packets (we use the number 100 in our simulation experiments).

Once the simulation of random vectors is completed, all the redundant faults (which include the random-resistant hard-to-detect faults) are removed from the overall fault list of the circuit and stored in a separate fault list. Since all of the faults remaining in the circuit's fault list are detectable, the goal of our approximate global fault technique becomes to eliminate faults from the list using equivalence and dominance relationships. We introduce the notation of a fault pair $\langle f_i, f_j \rangle$ to identify the relationship between faults f_i and f_j . It is obvious that if we have n faults in the fault list, then we have $n(n-1)/2$ fault pairs.

The type of the fault pair $\langle f_i, f_j \rangle$ can be any of the following:

- Equivalent (e) if f_i is equivalent to f_j .
- First dominating (f) if f_i dominates f_j .
- Second dominating (s) if f_i is dominated by f_j .
- Independent (i) if there is no relationship between f_i and f_j .

Initially, all fault pairs are of the equivalent type (e). As random test vectors are applied, the types of fault pairs are updated. The process is repeated for several iterations until no change is reported in the fault-pair types for a constant number of random packets (we use the number 100 in our simulation experiments). Figure 4 shows that state diagram describing the possible changes of fault-pair types. After applying a random vector t to a fault pair of type equivalent (e), the type of the fault pair does not change if (1) t does not detect any of the two faults in the fault pair or (2) t detects both faults in the fault pair. However, if t detects f_i but not f_j , then the type of the fault pair becomes first dominating (f). Finally, if t detects f_j but not f_i , then the type of the fault pair becomes second dominating (s). Similarly, the transitions from other fault-pair types in Figure 4 can be easily explained. It is interesting to notice

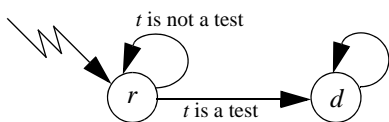


Figure 3 State diagram for tracking types of faults.

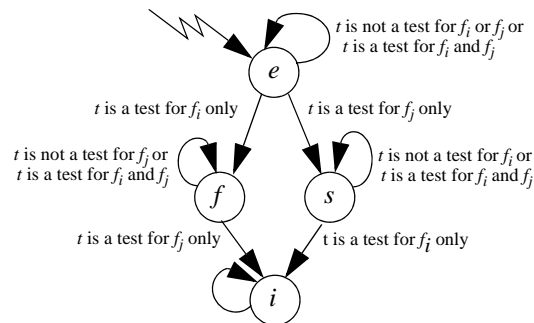


Figure 4 State diagram for tracking types of fault pairs.

that once the type of a fault pair becomes independent (i), it remains in that type forever.

The number of fault pairs with type (e) decreases as more random vectors are applied. On the other hand, the number of fault pairs with type (i) increases as more random vectors are applied. The number of fault pairs with types (f) or (s) often increases at the beginning but later decreases as more vectors are applied.

Once the simulation of random vectors is completed, the types of fault pairs are used to eliminate faults from the fault list according to the following rules: (1) If the type of the fault pair $\langle f_i, f_j \rangle$ is equivalent (e), then eliminate either f_i or f_j ; (2) If the type of the fault pair is first dominating (f), then eliminate f_j ; (3) If the type of the fault pair is second dominating (s), then eliminate f_i . Note that all fault pairs with independent (i) type are discarded since they do not help in eliminating faults from the fault list.

After the elimination of equivalent and dominating faults from the fault list as described above, we obtain an approximate global collapsed fault list. Now, we have to choose what to do with the (likely) redundant faults that we have extracted earlier. We have two possible alternatives:

- Option 1: Discard the likely redundant faults. This will speed up the test generation since no time is wasted in targeting the redundant faults. Moreover, if the likely redundant faults are in fact redundant, the fault coverage is not affected. On the other hand, if the likely redundant faults are in fact random-resistant hard-to-detect faults, then the coverage will be reduced by a little percentage (that is often negligible).
- Option 2: Add the likely redundant faults to the approximate global collapsed fault list. This will slow down the test generation but the fault coverage will not be affected. If the majority of the likely redundant faults are in fact redundant, then plenty of test generation time is often wasted.

Since the goal is to produce an approximate global collapsed fault list that will simplify the test generation and since the loss in fault coverage is often negligible if we discard the likely redundant faults, we adopt option 1 in this paper. However, option 2 can be used if the need arises.

Since the number of fault pairs is proportional to the square of the number of faults, then the storage of the

fault pairs in memory become prohibitive for large circuits. To solve this problem, we first simulate a random packet and then decide on the types of fault pairs. This allow us to skip storing fault pairs of type (*i*) in memory since they are not useful in fault collapsing. Moreover, during the simulation, fault pairs that change their type to (*i*) are automatically removed from memory. It should be noted that most fault pairs will ultimately be identified as of type (*i*). For example, out of 29,718,195 possible fault pairs in the ISCAS-85 circuit c6288 [3], there are at least 29,658,213 fault pairs of type (*i*). So, only 0.2% of the fault pairs are useful in determining the collapsed fault list and this is the same set of fault pairs that remain in memory at the end of the simulation.

In addition to the above, we can further limit the memory needed for the storage of fault pairs by enforcing a maximum value on the number of fault pairs that can be stored in memory. Once the fault pairs are initiated, they are stored in memory until the maximum value is reached. We then simulate a random packet to identify fault pairs of type (*i*) that are residing in memory. Consequently, these fault pairs are removed from memory and a new batch of fault pairs are initiated that can be stored in memory. The process is repeated until all fault pairs are processed.

Our AGFC tool implements the techniques presented in this section for the elimination of faults from the fault list. AGFC is written using C++ in approximately 5600 lines of code. A detailed description of the AGFC algorithm as well as experimental results that illustrate its capabilities can be found in [9].

3 EXACT GLOBAL FAULT COLLAPSING: A HYBRID APPROACH (EGFC-HYB)

In this section, we describe our new method of exact global fault collapsing using binary decision diagrams and fault simulation. In this method, we first apply a large set of random vectors in order to identify the likely redundant faults similar to AGFC. We then use the method of EGFC-BDD to prove whether the faults are redundant or not. Consequently, all the proven redundant faults are removed from the overall fault list of the circuit.

Since all of the faults remaining in the circuit's fault list are detectable, then our next goal is to remove faults from the fault list using equivalence and redundancy. Similar to AGFC, we apply random vectors to identify the types of fault pairs. At the end of the simulation, fault pairs are identified as either of type equivalent (*e*), first dominating (*f*), second dominating (*s*), or independent (*i*). However, the resulting assignment of types to the fault pairs is in fact an approximation since not all possible vectors are applied. Table 1 shows the relationship between approximate and actual fault pair

Table 1 Relationship between approximate and actual fault pair types.

Approximate fault pair type	(<i>e</i>)	(<i>f</i>)	(<i>s</i>)	(<i>i</i>)
Corresponding possible actual type of the fault pair	(<i>f</i>), (<i>s</i>), or (<i>i</i>)	(<i>f</i>) or (<i>i</i>)	(<i>s</i>) or (<i>i</i>)	(<i>i</i>)

types. This table can be easily derived from Figure 4. After the simulation of random vectors, fault pairs that are assigned a type (*i*) are in fact proven to be of type (*i*). However, this does not apply to other types of fault pairs. Since our goal in the hybrid method is to find the exact global collapsed fault list, then before using a fault pair of type (*e*), (*f*), or (*s*) to remove faults from the fault list, we need to prove or correct the fault pair type, using binary decision diagrams as used in EGFC-BDD. Figure 5 shows the algorithm used by our EGFC-HYB method. It is obvious that the method would benefit significantly by reducing the number of fault pairs of type (*e*), (*f*), or (*s*). The less the number of fault pairs that need to be identified using ROBDDs, the faster the execution of EGFC-HYB method.

Our EGFC-HYB tool implements the techniques presented in this section for the elimination of faults from the fault list. It is written using C++ in approximately 9000 lines of code.

We now describe experimental results that illustrate the capabilities of EGFC-HYB. The combinational library modules used in the experiments are the ISCAS-85 benchmark circuit c17 [3], 2-input MUX, 4-input MUX, a 3-input majority circuit, a circuit r10 with few redundant faults, as well as three circuits from the 74X TTL IC series [11]. The exact global fault collapsing results for the considered modules are shown in Table 2. The first column in the table reports the library module name. The next column reports the possible number of SSL faults (twice the number of lines in the circuit). The next column reports the number of SSL faults reported by the netlist which is often the local collapsed fault list of the circuit. The next column reports the number of random packets used by EGFC-HYB. It should be clear that the number of packets should be at least 100 since this is the condition needed to stop the simulation. The next column reports the number of

```

/* C is the circuit*/
procedure EGFC-HYB(C);
begin
  Form the fault/error list L
  /* Redundancy Identification */
  repeat
    Select a random packet P of 32 tests
    Fault simulate using P on L
    Update fault types (redundant or detectable)
  until no change is reported in 100 random packets or
    all faults are detectable
  Remove faults with redundant type that are proven to be
    redundant using ROBDDs
  /* Equivalence and dominance relationship identification
  Build fault pair list
  repeat
    Select a random packet P of 32 tests
    Fault simulate using P on L
    Update types of fault pairs
  until no change is reported in 100 random packets
  repeat
    Select a fault pair FP not of type (i)
    Determine the type of FP using ROBDDs
    Remove a fault f from fault list according to rules
    Remove all fault pairs using f
  Until fault pair list is empty.
  Output the results;
end;

```

Figure 5 EGFC-HYB's algorithm.

Table 2 Exact global fault collapsing results for the considered modules using EGFC-HYB.

Combinational library module	Possible SSL faults	Netlist SSL faults	No. of random packets	No. of redundant faults	No. of fault pairs at end of simulation				Verified pairs using BDDs	Fault list		Execution time
					(e)	(f)	(s)	(i)		Size	%	
c17	34	22	103	0	1	18	25	187	11	11	32.36	2.63
2-input multiplexer	18	18	102	0	13	6	22	112	14	4	22.22	2.46
4-input multiplexer	46	46	106	0	38	54	106	837	34	12	26.09	5.26
3-input majority circuit	26	26	102	0	18	24	54	229	20	6	23.08	3.38
r10 (redundant circuit)	20	20	202	2	34	0	15	104	14	4	20.00	3.23
7485 (4-bit comparator)	234	137	143	0	12	362	406	8536	89	48	20.51	33.03
74181 (4-bit ALU)	398	237	158	0	26	709	1096	26135	156	81	20.35	439.13
74283 (4-bit adder)	208	128	118	0	22	373	524	7209	104	24	11.54	25.19

proven redundant faults reported by EGFC-HYB. The next four columns in the table report the number of fault pairs at the end of the simulation in each of the four fault pair types. The next column in the table report the number of fault pairs that was verified (using ROBDDs) by EGFC-HYB. Note that this number is significantly less than the total number of fault pairs with types (e), (f), and (s). The next two columns in the table report the exact global collapsed fault list size (and percentage from total faults) produced by EGFC-HYB. The next column in the table shows the execution time in seconds for a sample run on a Dell computer (2.2 GHz Pentium II, 512MB of RAM) running Windows XP. It should be noted that the time varies from one run to another since it depends on the quality of the generated random vectors.

Our EGFC-HYB tool was not able to compute the exact global collapsed fault list for larger ISCAS-85 benchmark circuits since the memory needed for building the ROBDDs is more than the physical memory of the used workstation. However, more results can be obtained using a better machine (with larger physical memory and faster processor).

4 COMPARISONS AND EXTENSIONS

EGFC-BDD is an exact global fault collapsing tool for combinational circuits. It computes the tests for a given fault using ROBDDs and then eliminates faults using redundance, equivalence, and dominance. The experiments reported in [8] show a number of interesting observations: (1) EGFC-BDD is relatively fast, (2) EGFC-BDD produced a small collapsed fault list; and (3) the ratio of the EGFC-BDD fault list to the total number of circuit faults can be as low as 11.54%.

AGFC is an approximate global fault collapsing tool for combinational circuits. It is based on a novel combination of parallel-pattern evaluation, multiple fault activation, single fault propagation, critical path tracing, and random simulation. The experiments reported in [9] show that AGFC is relatively fast. They also confirm a number of interesting observations such as: (1) AGFC produced a small collapsed fault list that matches the exact global collapsed fault list for small library modules, and (2) it produced better results in comparison to prior work.

EGFC-HYB is an exact global fault collapsing method that is based on the integration of fault simulation and binary decision diagrams. It uses a combina-

tion of the features of EGFC-BDD and AGFC in order to achieve a fast global fault collapsing.

It should be noted that our tools are preliminary implementations that can be further improved in order to quickly handle larger library modules. Improving the implementation of BDD manipulations will definitely lead to less memory and execution time and consequently produce fault collapsing results for the large ISCAS-85 circuits.

By comparing the three presented global fault collapsing methods, we conclude the following:

- AGFC is an approximate global fault collapsing method. So, we cannot measure how close is the produced approximate collapsed fault list to the exact collapsed fault list. However, we can guarantee that the size of the approximate global collapsed fault list is less than or equal to the size of the exact global collapsed fault list. Nevertheless, the method can handle large library modules (such as the largest ISCAS-85 benchmark circuits) and in fact it produced the exact global collapsed fault list for the small library modules used in the experiments.
- EGFC-BDD is an exact global fault collapsing method. The produced fault list is the smallest possible fault list size. The drawback of this method is its high demand of memory and execution time.
- EGFC-HYB is a faster version of EGFC-BDD. It uses the method of AGFC to reduce the amount of computations needed for ROBDDs. However, the memory requirements for EGFC-HYB remain the same as that of EGFC-BDD.

To illustrate the comparison among the three presented fault collapsing methods, consider a library module M that uses the circuit 74181 ALU. M 's characteristics and the summary of the fault simulation results for each of the three methods are shown in Figure 6. We can conclude from this figure that AGFC is 67 times faster than EGFC-HYB and 1197 times faster than EGFC-BDD for the module M . However, the collapsed fault list produced by AGFC is 2 faults less than the exact global collapsed fault list of M . EGFC-HYB and EGFC-BDD produced the same exact global fault collapsing results of M , but with EGFC-HYB being 18 times faster than EGFC-BDD.

When library modules do become large in size, the process of exact global collapsing of faults eventually becomes tedious and time consuming even if we have unlimited memory. A method for expediting the pro-

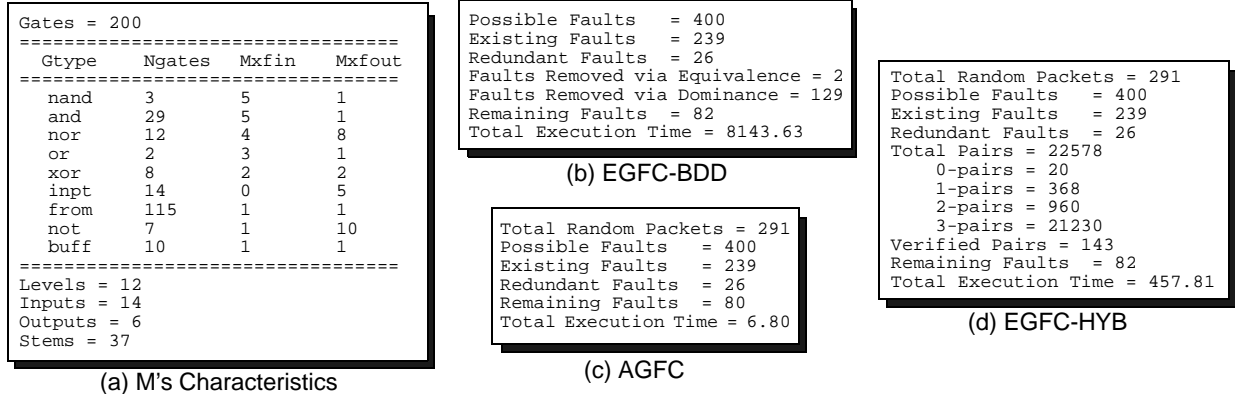


Figure 6 (a) M 's characteristics & summary of fault simulation results for (b) EGFC-BDD, (c) AGFC, & (d) EGFC-HYB.

cess of global fault collapsing is to take the middle ground between global and local fault collapsing. In this process, we take a complex library module and partition it into smaller modular components. We then perform global fault collapsing for each of the components using EGFC-HYB so that we end up with a list of the remaining faults that characterize each of the components. Note that we can perform the exact global fault collapsing on the components in parallel on different computers and hence produce the fault collapsing results faster. We finally combine the collapsed fault lists of components to produce an approximation of the globally collapsed fault list of the overall module.

To illustrate the advantage of partitioning the design, assume that an arbitrary large module is partitioned into k components each with n/k faults, where n is the number faults in the large module. Instead of analyzing $n(n-1)/2$ fault pairs, AGFC or EGFC-HYB needs to analyze $(n/k)((n/k)-1)/2$ fault pairs for each module. So, the overall number of fault pairs that need to be analyzed is approximately the total number of fault pairs divided by k^2 . Hence, a significant speedup in fault collapsing can be achieved as a result of partitioning the design.

It may be the case that the execution time of AGFC, EGFC-BDD, or EGFC-HYB is longer than the ATPG applied on a non collapsed fault list. So, where is the advantage of using the fault collapsing? The collapsed fault list is computed once for every library module and it can be stored with the module in the design library. For large designs with multiple modules from the design library, the faults of the modules can be added together to form the fault list of the overall design. ATPG execution time for large designs with multiple modules will eventually surpass the execution time of fault collapsing for the individual modules.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0092867.

REFERENCES

- [1] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*,

IEEE Press, New Jersey, 1994.

- [2] A. Lioy, "Advanced fault collapsing", *IEEE Design and Test of Computers*, Vol. 9, pp. 64-71, January 1992.
- [3] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran", *Proc. International Symposium on Circuits and Systems*, 1985, pp. 695-698.
- [4] A. V. S. S. Prasad, V. D. Agrawal, and M. V. Atre, "A new algorithm for global fault collapsing into equivalence and dominance sets", *Proc. International Test Conference*, 2002, pp. 391-397.
- [5] V. D. Agrawal, A. V. S. S. Prasad, and M. V. Atre, "Fault collapsing via functional dominance", *Proc. International Test Conference*, 2003, pp. 274-280.
- [6] M. E. Amyeen et al., "Fault equivalence identification in combinational circuits using implication and evaluation techniques", *IEEE Transactions on CAD*, Vol. 22, pp. 922-936, July 2003.
- [7] H. Al-Asaad and R. Lee, "Simulation-based approximate global fault collapsing", *Proc. International Conference on VLSI*, 2002, pp. 72-77.
- [8] H. Al-Asaad, "EGFC: An exact global fault collapsing tool for combinational circuits", *Proc. IASTED conference on Circuits, Signals, and Systems*, 2005, pp. 56-61.
- [9] H. Al-Asaad, "AGFC: An approximate simulation-based global fault collapsing tool for combinational circuits", *Proc. International conference on Circuits, Signals, and Systems*, 2006, pp. 248-253.
- [10] R. Bryant, "Graph-based algorithms for boolean function manipulation", *IEEE Transactions on Computers*, Vol. C-35, pp. 677-691, August 1986.
- [11] Texas Instruments, *The TTL Logic Data Book*, Dallas, 1988.
- [12] R. Sandireddy and V. D. Agrawal, "Diagnostic and detection fault collapsing for multiple output circuits", *Proc. Design, Automation and Test in Europe*, 2005, pp. 1014-1019.
- [13] H. Al-Asaad and J. P. Hayes, "ESIM: A multimodel design error and fault simulator for logic circuits", *Proc. IEEE VLSI Test Symposium*, 2000, pp. 221-228.