EGFC: AN EXACT GLOBAL FAULT COLLAPSING TOOL FOR COMBINATIONAL CIRCUITS

Hussain Al-Asaad
Department of Electrical & Computer Engineering
University of California
One Shields Avenue, Davis, CA 95616-5294
E-mail: halasaad@ece.ucdavis.edu

Abstract

Fault collapsing is the process of reducing the number of faults by using redundance and equivalence/dominance relationships among faults. Exact fault collapsing can be easily applied locally at the logic gates, however, it is often ignored for most circuits, due to its high demand of resources such as execution time and/or memory. In this paper, we present EGFC, an exact global fault collapsing tool for combinational circuits. EGFC uses binary decision diagrams to compute the tests for faults and consequently achieve efficient global fault collapsing. Experimental results show that EGFC reduces the number of faults drastically with feasible resources.

Keywords: Global fault collapsing, fault simulation, physical fault testing.

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 redundance, 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 most 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.

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.

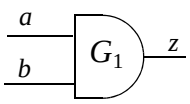
In this paper, we describe EGFC, an efficient tool for exact global fault collapsing. We review fault collapsing in Section 2 and present our efficient technique for exact fault collapsing and the EGFC tool in Section 3. We present some experimental results in Section 4 and finish with our conclusions in Section 5.

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*.



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

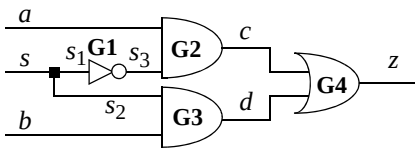
Figure 1 Fault collapsing for a 2-input AND gate.

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 consequently 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 redundance, 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 are 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 redundance, equivalence and dominance relationships among the faults in the overall circuit. In the next section, we describe our technique for exact global fault collapsing and the EGFC tool.



Gate G1: {s_3/0, s_3/1}
Gate G2: {a/1, s_3/1, c/0}
Gate G3: {b/1, s_2/1, d/0}
Gate G4: {c/0, d/0, z/1}
Stem s: {s/0, s/1}

Inputs			Correct	Faulty functions									
s	a	b	output z	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.

3 Exact Global Fault Collapsing

In this section, we determine the conditions needed to establish redundance, 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 $X = x_{n-1}...x_1x_0$. Also, let the m outputs of C be represented as $Y = y_{m-1}...y_1y_0$. 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 in Section 2). However, 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:

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

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:

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

Hence, f is equivalent to g. Based on the above, we need to compute the two functions $D_{fg}(X, f, g)$ and $D_{gf}(X, f, g)$ and 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) [8] 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 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 can be easily modified so that it uses less memory on the expense of more execution time.

Our EGFC tool implements the techniques presented above for the elimination of faults from the fault list. EGFC is written using C++ in approximately 7000 lines of code. A simplified view of its main algorithm is shown in Figure 3.

4 Experimental Results

We now describe experimental results that illustrate the capabilities of EGFC. The circuits 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 [9]. A sample run of the EGFC tool on the 74283 4-input ripple carry adder is shown in Figure 4.

The exact global fault collapsing results for the above considered circuits are shown in Table 1. The first column in the table reports the circuit name. The next column reports the possible number of SSL faults (twice the number of lines in the circuit). The next two columns report the number of inputs and outputs in the circuit, respectively. 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 three columns report the number of faults removed via redundance, equivalence,

```
/* C is the circuit*/
procedure EGFC(C);
begin
   Build fault-free ROBDDs
   Form the fault/error list L
   /* Redundancy Identification */
   for every fault f in fault list L
   begin
      Determine if f is redundant (using ROBDDs)
      Remove f from fault list L if it is redundant
   end
   /* Equivalence and dominance relationship identification
   for every fault f in fault list L
   begin
      for every fault g in fault list L
      begin
         Using ROBDDs, determine C1 = (f dominates g)
         Using ROBDDs, determine C2 = (g dominates f)
         if C1 = true and C2 = true then
         begin
            The faults f and g are equivalent
            Mark f for deletion
         end
         else if C1 = true then Mark f for deletion
         else if C2 = true then Mark g for deletion
      end
   end
   Remove marked faults from fault list
   Output the results
end;
```

Figure 3 EGFC main algorithm.

Table 1 Exact global fault collapsing results for the considered circuits.

Circuit	Possible SSL faults	Circuit characteristics		Netlist SSL faults	Faults removed via redundance	Faults removed via equivalence	Faults removed via dominance	EGFC collapsed fault list		EGFC execution time
		Inputs	Outputs					Size	%	
c17	34	5	2	22	0	0	11	11	32.36	0.64
2-input multiplexer	18	3	1	18	0	6	8	4	22.22	0.16
4-input multiplexer	46	6	1	46	0	15	19	12	26.09	1.59
3-input majority circuit	26	3	1	26	0	9	11	6	23.08	0.29
r10 (redundant circuit)	20	3	1	20	2	10	4	4	20.00	0.12
7485 (4-bit comparator)	234	11	3	137	0	10	79	48	20.51	654.83
74181 (4-bit ALU)	398	14	8	237	0	0	156	81	20.35	17101.61
74283 (4-bit adder)	208	9	5	128	0	12	92	24	11.54	140.21

```
EGFC Copyright 2005
Programmer: Hussain Al-Asaad

Gates = 104
================================
 Gtype      Ngates      Mxfin      Mxfout
================================
 nand       4           2          7
 and        14          5          1
 nor        8           5          5
 or         0           0          0
 xor        4           2          0
 xnor       0           0          0
 inpt       9           0          2
 from       59          1          1
 not        6           1          5
 buff       0           0          0
 GND        0           0          0
 VDD        0           0          0
 dff        0           0          0
================================
Levels = 6
Inputs = 9
Outputs = 5
Stems = 22
Tests = 12

Redundant Fault Identification
==============================

Redundant Faults:

Equivalent and Dominant Fault Identification
====================================

Equivalent and Dominant Faults:

2inpt Stuck_at_0
......
102xor Stuck_at_0
103xor Stuck_at_0

Global Collapsed Fault List
===========================

Remaining Faults:

Gate: 6from    Fault Type:Stuck_at_0
Gate: 7from    Fault Type:Stuck_at_1
.........
Gate: 88and    Fault Type:Stuck_at_0

====================================
===========SUMMARY RESULTS=========
====================================
Possible Faults   = 208
Existing Faults   = 128
Redundant Faults  = 0
Faults Removed via Equivalence = 12
Faults Removed via Dominance = 92
Remaining Faults  = 24

Total Execution Time = 140.21
```

Figure 4 Output generated by a sample run of EGFC.

and dominance, respectively. The next two columns in the table report the exact global collapsed fault list size (and percentage from total faults) produced by EGFC. It should be noted that the size of the EGFC fault list can be as low as 11.54% of the total faults which corresponds to 88.46% reduction in the fault list. The final column in the table shows the execution time (in seconds) of EGFC for the given circuit using an HP 9000/785 Workstation (400MHz and 256MB).

Our EGFC 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 a large physical memory).

5 Conclusions

EGFC is a 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 here show a number of interesting observations: (1) EGFC is relatively fast, (2) EGFC produced a small collapsed fault list; and (3) the ratio of the EGFC fault list to the total number of circuit faults can be as low as 11.54%.

Several aspects of EGFC remain to be investigated. For example, we need to enhance the implementation of EGFC so that it needs less memory and execution time and consequently produce fault collapsing results for the large ISCAS-85 circuits.

When circuits do become huge in size, the process of exact global collapsing of faults eventually becomes tedious and time consuming. A method for expediting the process of global fault collapsing is to take the middle ground between global and local fault collapsing. In this hybrid process, we take a complex circuit and partition it into smaller modular components. We then perform global fault collapsing for each of the components using EGFC so that we end up with a list of the remaining faults that characterize each of the components. Once this is accomplished, we recombine the entire circuit and target all the collapsed fault lists from each of the components. The premise is that the combined collapsed fault lists of the components produces an approximation of the globally collapsed fault list.

One benefit of this process is time. We initially perform global fault collapsing on smaller simpler circuits (components) and this process can be performed in parallel on different computers. When the fault collapsing of every component is completed, then the overall collapsed fault list of the circuit can be constructed. Moreover, a library of components with their collapsed fault lists can be constructed so that it can be used for other designs.

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* (Computer Science Press, New York, 1990).

[2] A. Lioy, Advanced fault collapsing, *IEEE Design and Test of Computers*, *9*(1), 1992, 64-71.

[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, 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, 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, 274-280.

[6] M. E. Amyeen et al., Fault equivalence identification in combinational circuits using implication and evaluation techniques, *IEEE Transactions on CAD*, *22* (7), 2003, 922-936.

[7] H. Al-Asaad and R. lee, Simulation-based approximate global fault collapsing, *Proc. International Conference on VLSI*, 2002, 72-77.

[8] R. Bryant, Graph-based algorithms for boolean function manipulation, *IEEE Transactions on Computers*, *C-35*(8), 1986, 677-691.

[9] Texas Instruments, *The TTL Logic Data Book* (Texas Instruments, Dallas, 1988).