Use of Hierarchy in Fault Collapsing

Raja K. K. R. Sandireddy*
Intel Corporation
Hillsboro, OR 97124, USA
srkkreddy@gmail.com

Vishwani D. Agrawal
Auburn University
Auburn, AL 36849, USA
vagrawal@eng.auburn.edu

Abstract

We discuss the advantage of using hierarchy in testing. Our demonstration is based on the problem of fault collapsing. Though this problem is not considered to be too complex, the time of collapsing faults in moderately large circuits can be several hours or more. This can be considerably shortened by hierarchical fault collapsing. Large circuits are efficiently described using hierarchy, which significantly helps the architectural design, verification and physical design. In hierarchical collapsing, we do not flatten the circuit and the collapsed fault sets computed once for sub-circuits are reused for all instances of those sub-circuits. The time required for collapsing a flattened 8192-bit ripple carry adder using a test program like Hitec is 84 minutes, while the fault collapsing using multiple levels of hierarchy takes 55 seconds. When functional functional fault techniques are used for smaller sub-circuits, hierarchical fault collapsing results in collapse ratios lower than those obtained with structural collapsing of flattened circuits, without compromising on the reduction in the CPU time for collapsing. Using functional collapsing for a full adder library cell, we hierarchically collapse faults in a 8192-bit adder to sets of 196,610 equivalence and 98,304 dominance collapsed faults. In comparison, the flattened circuit collapses into 294,914 and 229,378 equivalence and dominance collapsed sets, respectively. The advantage of smaller collapse ratios may be in the reduction of fault simulation effort and in the number of test vectors. It has been observed that the CPU time for structural fault collapsing for Boolean circuit by conventional programs grows as the square of the circuit size. A closer to linear time complexity can be expected for hierarchical fault collapsing.

1. Introduction

Fault Collapsing is widely used to reduce the number of target faults for test generation and fault simulation. It has been shown [13] that the cost of test generation has higher complexity than a linear relation with the number of faults to be tested. Hence, any technique reducing the size of collapsed fault set would improve the performance of test development procedure for VLSI circuits. Structural fault collapsing has been dealt thoroughly in any text book on digital testing [1, 8].

Fault collapsing can be classified into two types - equivalence collapsing and dominance collapsing. Two faults are called equivalent if and only if the corresponding faulty circuits have identical output functions [8]. Equivalent faults are indistinguishable because they cannot be distinguished at the primary outputs by any input vector. The set of all faults in a circuit can be partitioned into equivalence sets, such that all faults in a set are equivalent to each other. Equivalence collapsing is the process of fault partitioning and selecting one fault from each equivalence set. The fault set thus obtained is called equivalence collapsed set.

Another form of collapsing that can further reduce the fault set size is dominance fault collapsing. If all tests of fault f_1 detect another fault f_2, then f_2 is said to dominate f_1. The two faults are also called "conditionally" equivalent with respect to the test set of f_1 [8]. In dominance collapsing, all dominating faults in an equivalence collapsed set are left out retaining their respective dominated faults.

In Section 2, we discuss the background of fault collapsing. The hierarchical fault collapsing technique is described in Section 3. The results obtained with this technique are discussed in Section 4. A recent paper [35] describes some work presented here. More details are available in a thesis [34].

2. Background

There has been considerable work in the area of fault collapsing. Several authors [10, 16, 18, 25, 28, 36, 41] concentrate on finding the fault equivalences, while others [3, 26, 32] deal with fault dominance relations. Recent papers also give methods to find fault equivalences using ATPG [16, 41] and simulation [5]. Fault equivalence identification can be based on redundancy information [6], and hence test generation can prove equivalence [18]. There are techniques to find the relations between the faults on a fan-out stem, its branches and the reconvergent points [2, 27, 30, 38, 40].

It can be shown [14, 20] that identifying fault equiva-

*Formerly with Auburn University, Department of ECE, Auburn, AL 36849, USA, where this work was performed.

lence between two arbitrary faults in a combinational circuit is an NP-complete problem because it requires proving the equivalence of the two faulty functions. So, functional collapsing is not recommended for large circuits. Hence, most ATPG programs [9, 21, 23, 31] use only structural collapsing. Hierarchical fault collapsing seems to be an alternative that allows some functional collapsing. Hierarchical fault collapsing results in a collapse ratio lower than that obtained by structural collapsing alone, but not as low as obtainable if a complete functional collapsing could be used.

The increasing complexity of the circuits can be efficiently handled using a hierarchical design process. The hierarchical description of a circuit at the highest level consists of an interconnection of few blocks, which are described at a lower level of hierarchy as interconnects of other blocks and gates [43]. A block or a sub-circuit C_j in a circuit C_i is called an instance of C_j. The circuit defined by hierarchical description can be obtained by an expansion process referred to as flattening.

The hierarchy of a circuit description can be used in test generation [24, 42, 44] and fault simulation [22, 29, 33]. To the best of the author's knowledge, there have been few papers [3, 17, 32, 35] that have used hierarchy of the circuit to collapse faults. There are advantages of using hierarchy to collapse faults. A reduced fault set, that is computed once for a sub-circuit, can be reused for all instances of the sub-circuit. For smaller circuits, functional fault collapsing techniques can achieve better collapse ratios.

2.1. Graph Model

We use a graph model described in the literature [4, 32]. The fault equivalence and dominance relations are represented by a directed graph. In this graph each fault is represented by a node. If fault f_1 dominates fault f_2 then this is represented by a directed edge from node f_2 to f_1. This edge indicates that any test for f_2 must detect f_1. Clearly, the presence of edges $f_1 \rightarrow f_2$ and $f_2 \rightarrow f_1$ indicates that the two faults f_1 and f_2 are equivalent. Fault dominance graph, or simply a dominance graph, represents the dominance relations among the faults of a circuit.

Figure 1 shows the dominance graph for all faults of an OR gate. The subscript fault notation has been used, that is, a_0 means that the fault is on line named 'a' and is s-a-0. The dominance graph is conveniently represented by its dominance matrix shown in Table 1. A 1 entry at the intersection of a row and a column means that the fault corresponding to the column dominates the fault corresponding to the row. For example, the 1 in the second row and the last column indicates that c_1 dominates a_1. Equivalence of two faults is expressed by two 1's placed at both intersections of the rows and columns of those faults. Since there is also a 1 in the last row and second column indicating that a_1 dominates c_1, it can be said that a_1 and c_1 are equivalent. This dominance matrix is used in hierarchical fault collapsing technique to represent all the dominance relations between the faults.

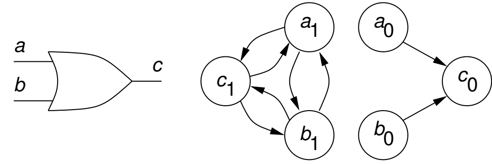

Figure 1. Dominance graph of an OR gate.

Table 1. Dominance matrix of OR gate.

	a_0	a_1	b_0	b_1	c_0	c_1
a_0	1	0	0	0	1	0
a_1	0	1	0	1	0	1
b_0	0	0	1	0	1	0
b_1	0	1	0	1	0	1
c_0	0	0	0	0	1	0
c_1	0	1	0	1	0	1

3. Hierarchical Fault Collapsing

Typically, faults in a hierarchically described circuit are collapsed after flattening the hierarchy [9]. In our method, we do not flatten the circuit to collapse the faults. Hierarchical fault collapsing is based on the following theorem [15, 17]:

Theorem: *If two faults are functionally equivalent in a combinational sub-circuit M that is embedded in a circuit N, then they are also functionally equivalent in N.* ■

Functional equivalence here means diagnostic equivalence, as defined in a recent paper [35]. Such faults are called *intrinsically equivalent* in M by Goundan and Hayes [15]. This is demonstrated using the circuit in Figure 2, where the s-a-1 faults on lines x and y are intrinsically equivalent in M. There is another kind of equivalence, called extrinsic equivalence [15]. If two faults f_i and f_j located in a sub-circuit, M, are equivalent in the circuit N, but are not equivalent in the sub-circuit, then they are called *extrinsically equivalent* in M. In Figure 2, the s-a-1 faults on lines p and s are not equivalent in sub-circuit M but become equivalent in the complete circuit. The extrinsically equivalent fault pairs form a subset of functionally equivalent fault pairs. Such relationships are not detected by hierarchical fault collapsing.

The collapsing starts at the topmost level of hierarchy. Structural equivalence collapsing is done at this level using a line oriented structural fault collapsing technique as explained by Nadjarbashi *et al.* [30]. Next, the dominance relations between the faults in the equivalence collapsed set are found using the algorithm in the next section. For every s-a-0 on any input of an OR or NOR gate, the fault among the equivalent set that dominates it is found by setting b as 0 in the algorithm. Then a 1 is introduced in the dominance matrix to indicate this dominance using the $update$ algorithm [11, 12], which computes the transitive closure of the dominance matrix. A similar procedure is followed for s-a-

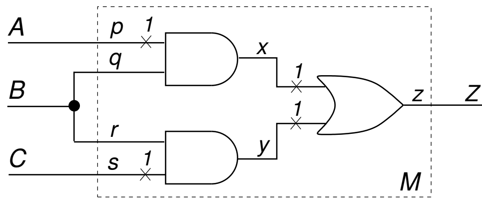

Figure 2. A circuit demonstrating extrinsic equivalence.

1 faults on the inputs of AND and NAND gates.

3.1 Algorithm to Find the Dominance Matrix and its Transitive Closure

1. Consider a stuck-at-b fault (f_1) in the equivalence collapsed set at the input of a Boolean gate.
2. If the gate is of inverting type (NOT, NOR, NAND), then invert b.
3. If the equivalent set has s-a-b on this gate output, say f_2, then place a 1 at the intersection of the row corresponding to f_1 and column corresponding to f_2 and use $update$ [11, 12] for computing transitive closure. Go to step 1 until all faults in the equivalence collapsed set that are on the inputs of Boolean gates are considered.
4. Move one gate forward toward the primary output and go to step 2.

Next, the processing of instances of the sub-circuits is done. If the sub-circuit has a library file containing its collapsed information, that file is used to introduce new faults and relations into the dominance matrix. If there is no such library file, a library file is created dynamically as follows. If the sub-circuit does not have any user-defined gates, namely, instances of other sub-circuits, in its description and has less than a prespecified number of gates, say 15, we can use the functional collapsing algorithms [35]. Otherwise, the sub-circuit is processed recursively in the same way as the top level description is processed. This continues until all sub-circuits are processed. The collapsed fault set description is written to a library file for future use.

To obtain the equivalence collapsed set, the fault set is processed by *algorithm equivalence* [32]. This step removes all extra faults that were added to assist the hierarchical fault collapsing. If dominance collapsing is required, then *algorithm dominance* [32] is used.

4. Results

In this section, we present the results of hierarchical fault collapsing, comparing the collapse ratios and collapsing time. The library consists of files containing the collapsed information of frequently used circuits like multiplexer, XOR, half adder, full adder, etc. Since the dominance matrix has very few 1's, it is implemented using a sparse matrix representation [19]. The transitive closure is implemented using the *update* algorithm as described by Dave *et al.* [11, 12]. This algorithm has a linear time complexity in circuit size for sparse matrices, though in the worst case its complexity remains $O(n^3)$, where n is the number of faults.

4.1 Comparison of Collapse Ratios

The collapsed fault set sizes obtained with hierarchical fault collapsing for different circuits are shown in Table 2. Those are compared with structural collapsing results for the corresponding flattened circuits obtained using Hitec [31] and Fastest [21] programs. The 4-bit adder consists of four full adders, the 16-bit adder has four 4-bit adders, and so on. The full adder circuit shown in the table is not a hierarchical circuit and the collapsing results of this circuit shown under the column Hierarchical are functional (diagnostic) collapsing results [35]. The ISCAS'85 circuits [7] c432 and c499 have XOR gates which are considered as user-defined gates.

The collapse ratio of each collapsed set is shown by the accompanying fraction in parenthesis. There is considerable improvement in the collapse ratios when hierarchical collapsing is used. The reduction achieved for XOR and full adder sub-circuits using functional collapsing is passed onto other levels because of hierarchical collapsing. Even if all hierarchical adder circuits are described using only one-bit full adders, for example, a 1024-bit hierarchical adder is built using 1024 one-bit full adder sub-circuits, the collapse ratios would be the same. This is because, the reduction in these hierarchical adders is solely due to the functional collapsing done up to the full adder level. If the library files used for XOR and full adder circuits had structural, instead of functional, collapsed set information, the hierarchical fault collapsed sets would have the same collapse ratios as obtained for flattened circuits. Functional collapsing here refers to diagnostic collapsing [35]. If detection collapsing [35] were to be used in any hierarchical adder we could have used it on only one sub-circuit, the one which has all its outputs as primary outputs. This causes a reduction of 3 and 6 faults in any of the equivalence and dominance collapsed sets, respectively, under the column Hierarchical in Table 2.

4.2 Comparison of CPU Times of Fault Collapsing

We have shown that hierarchical collapsing results in better (lower) collapse ratios. Now, we show that the CPU time taken by hierarchical collapsing is also lower than that required for flattened (structural) collapsing. A C program implementing the procedure is used to collapse the faults in both flattened and hierarchical circuits. The CPU times reported in seconds in all the tables in this section is clocked on a 360MHz Sun UltraSparc 5_10 machine with 128MB memory.

First, the implementation efficiency of our program is checked. We used differently sized flattened adder circuits

Table 2. Collapsed fault sets.

Circuit name	Number of all faults	Number of collapsed faults (collapse ratio)			
		Flattened (structural)		Hierarchical (functional)	
		Equiv.	Dom.	Equiv.	Dom.
Full adder	60	38 (0.63)	30 (0.50)	26 (0.43)	12 (0.20)
4-bit adder	234	146 (0.62)	114 (0,49)	98 (0.42)	48 (0.21)
16-bit adder	930	578 (0.62)	450 (0.48)	386 (0.42)	192 (0.21)
64-bit adder	3714	2306 (0.62)	1794 (0.48)	1538 (0.41)	768 (0.21)
256-bit adder	14850	9218 (0.62)	7170 (0.48)	6146 (0.41)	3072 (0.21)
1024-bit adder	59394	36866 (0.62)	28674 (0.48)	24578 (0.41)	12288 (0.21)
8098-bit adder	475138	294914 (0.62)	229378 (0.48)	196610 (0.41)	98304 (0.21)
c432	1116	632 (0.56)	503 (0.45)	524 (0.47)	413 (0.37)
c499	2646	1574 (0.59)	1210 (0.46)	950 (0.36)	690 (0.26)

Table 3. Fault collapsing time for flattened circuits.

Circuit name	CPU time (seconds)		
	AUSIM	Hitec	Our Program
4-bit adder	0.08	0.23	0.02
8-bit adder	0.11	0.24	0.03
16-bit adder	0.20	0.30	0.04
32-bit adder	0.43	0.36	0.10
64-bit adder	1.10	0.57	0.24
128-bit adder	2.65	1.47	0.75
256-bit adder	9.60	5.09	2.49
512-bit adder	39.5	19.5	9.38
1024-bit adder	165	77.7	39.9
2048-bit adder	650	326	166.4
4096-bit adder	2623	1258	674.1
8192-bit adder	10681	–	2676

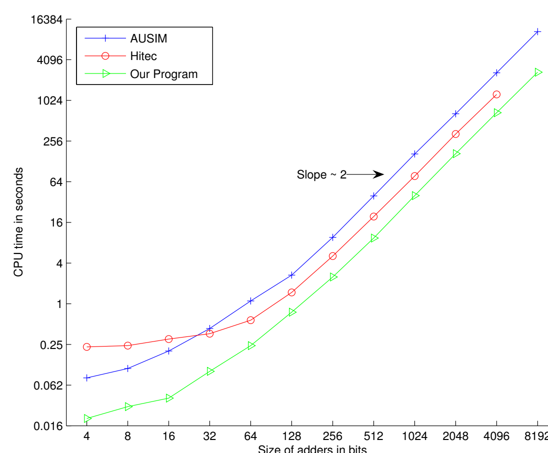

Figure 3. Fault collapsing time for flattened circuits.

(logic gate level circuits without any sub-circuits). The CPU time taken by our program for collapsing the faults is either similar to or better than that taken by other available programs [21, 31, 37]. In Table 3, we compare with the time taken by AUSIM [37] and Hitec [31]. Hitec and AUSIM do only equivalence collapsing while our program also provides dominance collapsed set. It should be noted that Hitec, in addition, calculates the controllabilities and observabilities for each line, which are later used in test generation. All programs resulted in the same equivalence collapsed set sizes for all circuits. We do not have the CPU time taken by Hitec for the flattened circuit of 8192-bit adder because it could not complete the collapsing operation for this circuit, probably due to some internal program limit on size of the circuit. It can be seen from the table that our program takes considerably less CPU time than the other two programs although it displays the same time complexity.

The CPU times shown in Table 3 are plotted on the graph in Figure 3. Both axes of the plot have logarithmic scales with base 2. The portion of the plot corresponding to adders smaller than the 64-bit adder is included here only for completeness and can be neglected because of the coarse resolution of the time commands used to record the CPU time. Since all the curves have a slope approximately equal to 2, it shows that the CPU time in all these cases is proportional to the square of the circuit size. This can be seen in Table 3 as well.

To understand the complexity of our program, we measured the time taken for different functions involved in collapsing. The time taken by different sections of our program for flattened circuits is tabulated in Table 4. The column Flat Structure Processing gives the time required for building the structure of circuit. The columns Equiv. and Dom. Collapsing show the time taken for collapsing faults based on structural equivalence and dominance, respectively. The total time taken for collapsing is shown in the last column of Table 4 and is same as the last column of Table 3.

When the structure of the circuit is built, any node representing a gate or primary input has links to all nodes in its fanin and fanout lists. The time taken to build such a structure is proportional to the square of the circuit size. This is confirmed by the time taken by our program to build the structure of the flattened circuits, as shown in the column Flat Structure Processing in Table 4. The time taken for equivalence and dominance collapsing grows linearly with

Table 4. CPU time of different sections of our program for flattened circuits.

Circuit name (adders)	CPU time (seconds)			
	Flat Structure Processing	Equiv. Collapsing	Dom. Collapsing	Total
4-bit	0.0	0.0	0.01	0.02
8-bit	0.0	0.0	0.01	0.03
16-bit	0.02	0.0	0.01	0.04
32-bit	0.05	0.01	0.01	0.10
64-bit	0.13	0.02	0.03	0.24
128-bit	0.54	0.05	0.05	0.75
256-bit	2.05	0.09	0.09	2.49
512-bit	8.60	0.17	0.20	9.38
1024-bit	38.3	0.36	0.41	39.9
2048-bit	163.1	0.74	0.84	166.4
4096-bit	667.4	1.49	1.63	674.1
8192-bit	2662	3.43	3.72	2676

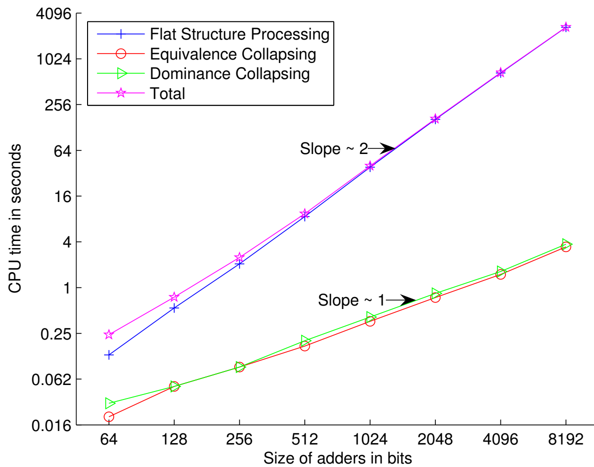

Figure 4. CPU time of different sections of our program for flattened circuits.

the circuit size. This is also verified using Figure 4, which is a plot of the data in Table 4 for 64-bit and larger adders. The slopes of the curves corresponding to Equivalence Collapsing and Dominance Collapsing are 1, while those of Flat Structure Processing and Total are 2. The plot also shows that, for large circuits, the total CPU time is dominated by the time needed to build the circuit structure. Hence, for large circuits, the total time required for collapsing grows as the square of the circuit size. Similar conclusions are drawn for the time for collapsing using AUSIM [37].

Next, our program is used to collapse faults in the circuits described hierarchically with many levels just like the hierarchical adder circuits in Table 2, i.e., a 1024-bit adder is built using four instances of a 256-bit adder, which is built using four instances of a 64-bit adder, and so on. The times for different functions in the program are shown in Table 5. The column Hierarchical Structure Processing gives the time taken to build the structure of the hierarchical circuit. The time required for both equivalence and dominance collapsing is shown under the column Equiv.+Dom. Collapsing, of which the time required for equivalence collapsing is the major component. The time taken to introduce new faults of the sub-circuits from library files and their relations is shown under the column Library. This time was negligible in Table 4, since there were no sub-circuits involved. The column Total gives the total time taken by the program which includes the time taken to dynamically collapse all the sub-circuits used in the hierarchical circuit. However, the one-bit full adder block which has been earlier collapsed using the functional technique is assumed to be available from a stored library. For example, the time shown against the 1024-bit adder includes the time of building the library files for 1024-bit, 256-bit, 64-bit, 16-bit and 4-bit adders, but not that of the 1-bit adder library element. It can be seen that, even here the time required to build the circuit (netlist structure processing) dominates as the size of the circuit grows.

Table 5. CPU time of different sections of our program for hierarchical circuits.

Circuit name (adders)	CPU time (seconds)			
	Structure Processing	Equiv.+Dom. Collapsing	Library	Total
4-bit	0.0	0.0	0.0	0.01
8-bit	0.0	0.0	0.01	0.02
16-bit	0.01	0.01	0.02	0.05
32-bit	0.01	0.01	0.03	0.07
64-bit	0.01	0.01	0.07	0.10
128-bit	0.03	0.02	0.13	0.24
256-bit	0.05	0.02	0.19	0.49
512-bit	0.17	0.04	0.36	1.05
1024-bit	0.55	0.08	0.73	2.31
2048-bit	2.10	0.20	1.52	4.80
4096-bit	9.25	0.37	3.1	16.6
8192-bit	40.1	0.79	6.0	55.0

We show the times taken by our program for hierarchical circuits and flattened circuits in Figure 5. Here we compare the times taken for building the structure and collapsing faults, though the collapse ratios obtained for hierarchical circuits are better (lower) than those for flattened circuits. The times taken by our program for flattened circuits are shown in solid line curves, while the times taken for hierarchical circuits are shown in dashed line curves. The collapsing time for flattened circuits is the sum of Equivalence and Dominance Collapsing columns of Table 4. The collapsing time for hierarchical circuits is the sum of Equiv.+Dom. Collapsing and Library columns of Table 5. It can be seen from Figure 5 that the time for collapsing in both cases grows linearly with the circuit size. But there is considerable improvement in the CPU time used to build the circuit. When we build the hierarchical structure, the internal nodes of sub-circuits are considered only once and at a lower level of hierarchy. But, in flattened circuits, the in-

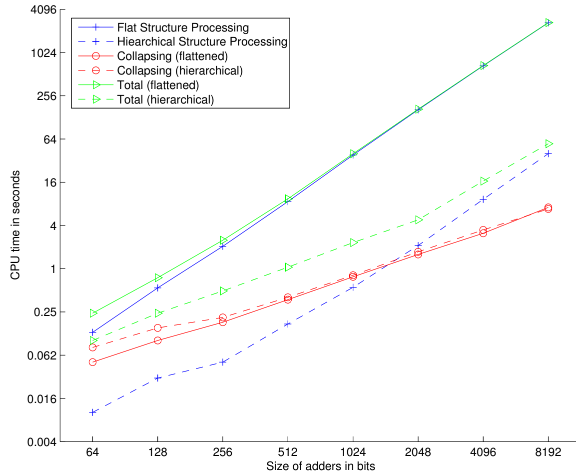

Figure 5. Comparison of CPU times of our program for hierarchical and flattened circuits.

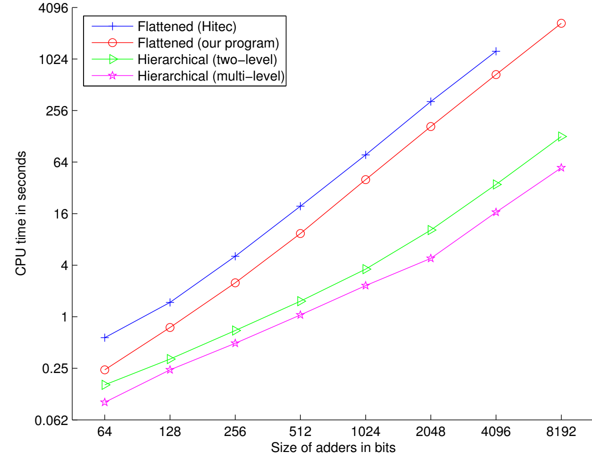

Figure 6. CPU time improvement by hierarchy.

ternal nodes of all the sub-circuits have to be considered at the same level. So, the number of nodes in the hierarchical structure is much smaller than that in the flat structure. This is the reason why the hierarchical structure is built much faster than the flat structure. This improvement shows up in the comparison of total times (curves drawn through triangles) for the two cases in Figure 5.

In Figure 5, the time to build the structure of a hierarchical circuit grows at a rate proportional to the square of the circuit size. It is expected that this complexity could be less than the square. This is because, in the circuits that we considered, different sized ripple carry adders, the number of inputs and outputs grow at the same rate as the circuit size. In general, however, according to Rent's rule [8, 39], the number of input and output terminals for a typical block containing G logic gates is given by Equation 1:

$$T = K \times G^{\alpha} \qquad (1)$$

where K is a constant between 1 and 5, and the exponent α lies in the range 0.5 to 0.67. For the ripple carry adders, the exponent α is close to 1.

The time to build the structure of the circuit is proportional to square of the number of gates and primary inputs in the circuit, as shown by the curves drawn with +'s in Figure 5. For a hierarchical circuit, the number of nodes that represent the structure of the circuit is dominated by the number of inputs and outputs, as the internal nodes of all sub-circuits are considered at a lower level of hierarchy. For a general hierarchical circuit, which obeys Rent's rule, the time taken to build the circuit should grow at a rate twice of α. If we assume $\alpha = 0.5$, then the time taken to build the circuit grows linearly with the circuit size. So, for a typical hierarchical circuit, the total time required for collapsing should have a near linear complexity in circuit size.

We now compare the CPU time required for fault collapsing with different levels of hierarchy. Table 6 shows

Table 6. CPU time improvement by hierarchy.

Circuit name	Flattened Circuit		Hierarchical Circuit	
	Hitec	Our Program	Two-level	Multi-level
4-bit	0.23	0.02	0.01	0.01
8-bit	0.24	0.03	0.02	0.02
16-bit	0.30	0.04	0.05	0.05
32-bit	0.36	0.10	0.08	0.07
64-bit	0.57	0.24	0.16	0.10
128-bit	1.47	0.75	0.32	0.24
256-bit	5.09	2.49	0.69	0.49
512-bit	19.5	9.38	1.52	1.05
1024-bit	77.7	39.9	3.60	2.31
2048-bit	326	166.4	10.3	4.80
4096-bit	1258	674.1	35.1	16.6
8192-bit	–	2676	127.2	55.0

three hierarchical versions of the adders. The values in the last column correspond to the circuits described hierarchically with many levels, just like the hierarchical circuits in Table 5, i.e., a 1024-bit adder is built using four 256-bit adders, and so on. The two-level circuits are built using a one-bit full adder as the sub-circuit, i.e., a 1024-bit adder is built using 1024 one-bit full adders. The flattened circuits can be seen as hierarchical circuits with a one-level hierarchy, i.e., they are described completely at the gate level.

As pointed out in Section 4.1 the collapse ratios obtained for the hierarchical circuits (i.e., two-level and multi-level) in Table 6 are the same. The data shown in Table 6 corresponding to 64-bit and larger adders is plotted in Figure 6. The time values in the last (multi-level) column include the time required to build the library file of collapsed sub-circuits at all levels. It can be observed from the figure that the circuits described using greater depth of hierarchy require less time for collapsing. When a different hierarchy was used, say a 1024-bit adder was built using eight 128-bit adders and the 128-bit adder used eight 16-bit adders, and so on, it resulted in similar times as reported in the last

Table 7. CPU times of our program for different levels of hierarchy.

Circuit name	CPU time (seconds)					
	1-bit	4-bit	16-bit	64-bit	256-bit	1024-bit
2048-bit adder	10.3	5.93	4.90	4.80	4.75	4.72
4096-bit adder	35.1	18.7	15.4	14.6	14.4	14.3
8192-bit adder	127.2	66.4	57.1	53.6	51.4	50.5

column. However, for another hierarchy when the 1024-bit adder was built using sixteen 64-bit adders, the 64-bit adder used sixteen 4-bit adders, and the 4-bit adder used four full adders, it required less time than the two-level circuit, but more than the multi-level circuit of Table 6.

If we assume that the library has the collapsing information for all sub-circuits used in the circuit, then the time required for collapsing monotonically decreases as the number of levels of hierarchy increases. But, when we have to build the collapsing information of the sub-circuits dynamically, we should take care that the time savings achieved by increasing the number of levels is not offset by the time required to collapse the sub-circuits. For example, when the library contains the collapsing information on 4096-bit and 2048-bit adders, a 8192-bit adder built using four 2048-bit adders is collapsed in 50.2 seconds, while a 8192-bit adder using two 4096-bit adders takes 50.0 seconds. The savings of 0.2 seconds is easily offset by the difference (11.8 seconds) between the times taken to build the library files for the sub-circuits, 4096-bit adder (16.6 seconds) and 2048-bit adder (4.8 seconds). So, when we have to dynamically build the library information of the sub-circuits, the time may monotonically decrease up to certain number of levels of hierarchy, depending upon the specific circuits.

As we increase the number of levels in the hierarchy, the number of nodes in the structure is decreased. But, the level to level reduction in the number of nodes slows down with increasing number of levels. So, the percentage saving in time decreases as the number of levels increase. This can be seen from Table 7 in which we give the collapse times of three circuits, 2048-bit, 4096-bit and 8192-bit adders. Each adder is built using only one type of sub-circuit, namely, 1-bit, 4-bit, 16-bit, 64-bit, 256-bit or 1024-bit adder. The time required for collapsing faults in each case is shown in Table 7. The columns in the table correspond to the sub-circuits used to build the circuit. For example, the 2048-bit adder takes 5.93 seconds when built using only 4-bit adders. It is assumed that the collapsing information of 1-bit, 4-bit, 16-bit, 64-bit, 256-bit and 1024-bit adders is present in the library. It is clearly seen that the reduction in time levels off as we move from left to right through the table.

5 Conclusion

In this paper, we have explained the hierarchical fault collapsing technique and the results obtained with this technique. We have shown that the benefit of smaller collapse ratios achieved by functional collapsing techniques for smaller sub-circuits is passed on to the larger hierarchical circuits. The advantage of such a small collapse ratio may be in the reduction of the fault simulation effort and in the number of test vectors. The time required for collapsing a hierarchical circuit is less than that for the corresponding flat circuit. The time taken for flat circuits is proportional to the square of the circuit size, while for hierarchical circuits, it is shown that the time for collapsing could be reduced closer to linear complexity. As the number of levels of hierarchy in the circuit increases, the time for collapsing decreases.

Care is needed in using dominance collapsed set since there can be instances where the dominated fault is redundant and the dominating fault (not included in the collapsed set) is testable. For fault diagnosis, we can only use the equivalence collapsed set.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*. Piscataway, New Jersey: IEEE Press, 1990.

[2] M. Abramovici, D. T. Miller, and R. K. Roy, "Dynamic Redundancy Identification in Automatic Test Generation," *IEEE Trans. Computer-Aided Design*, vol. 11, no. 3, pp. 404–407, Mar. 1992.

[3] V. D. Agrawal, A. V. S. S. Prasad, and M. V. Atre, "Fault Collapsing via Functional Dominance," *Proc. International Test Conf.*, 2003, pp. 274–280.

[4] S. B. Akers, C. Joseph, and B. Krishnamurthy, "On the Role of Independent Fault Sets in the Generation of Minimal Test Sets," *Proc. International Test Conf.*, 1987, pp. 1100–1107.

[5] H. Al-Asaad and R. Lee, "Simulation-Based Approximate Global Fault Collapsing," *Proc. International Conf. on VLSI*, 2002, pp. 72–77.

[6] M. E. Amyeen, W. K. Fuchs, I. Pomeranz, and V. Boppana, "Fault Equivalence Identification using Redundancy Information and Static and Dynamic Extraction," *Proc.* 19th *IEEE VLSI Test Symp.*, 2001, pp. 124–130.

[7] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuit and a target translation in FORTRAN," distributed on a tape to participants of the Special Session on ATPG and Fault Simulation, International Symposium on Circuits and Systems, June 1985; partially characterized in F. Brglez, P. Pownall, R. Hum, "Accelerated ATPG and Fault Grading via Testability Analysis," *Proc. IEEE International Symposium on Circuits and Systems*, June 1985, pp. 695-698, reprint of the article is available with the tape from MCNC. Website: http://www.cbl.ncsu.edu/CBL_Docs/Bench.html.

[8] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Boston, MA: Kluwer Academic Publishers, 2000.

[9] W. T. Cheng and T. J. Chakraborty, "Gentest: An Automatic Test Generation System for Sequential Circuits," *Computer*, vol. 22, no. 4, pp. 43–49, April 1989.

[10] F. W. Clegg, "Use of SPOOF's in the Analysis of Faulty Logic Networks," *IEEE Trans. Computers*, vol. 22, no. 3, pp. 229–234, Mar. 1973.

[11] K. K. Dave, "Using Contrapositive Rule to Enhance the Implication Graphs of Logic Circuits," Master's thesis, Rutgers University, New Brunswick, May 2004.

[12] K. K. Dave, V. D. Agrawal, and M. L. Bushnell, "Using Contrapositive Law in an Implication Graph to Identify Logic Redundancies," *Proc.* 18th *International Conf. VLSI Design*, Jan. 2005, pp. 723–729.

[13] P. Goel, "Test Generation Costs Analysis and Projections," *Proc.* 17th *IEEE/ACM Design Automation Conf.*, June 1980, pp. 77–84.

[14] A. Goundan, *Fault Equivalence in Logic Networks*. PhD thesis, University of Southern California, Los Angeles, Feb. 1978.

[15] A. Goundan and J. P. Hayes, "Identification of Equivalent Faults in Logic Networks," *IEEE Trans. Computers*, vol. C-29, no. 11, pp. 978–985, Nov. 1980.

[16] T. Grüning, U. Mahlsdedt, and H. Koopmeiners, "DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits," *Proc. International Conf. Computer-Aided Design*, Nov. 1991, pp. 194–197.

[17] R. Hahn, R. Krieger, and B. Becker, "A Hierarchical Approach to Fault Collapsing," *Proc. European Design & Test Conf.*, 1994, pp. 171–176.

[18] I. Hartanto, V. Boppana, and W. K. Fuchs, "Diagnostic Fault Equivalence Identification Using Redundancy Information & Structural Analysis," *Proc. International Test Conf.*, Oct. 1996, pp. 294–302.

[19] E. Horowitz and S. Sahni, *Fundamentals of Data Structures in PASCAL*. New York, USA: Computer Science Press, Inc, 1984.

[20] O. H. Ibarra and S. K. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Trans. Computers*, vol. C-24, pp. 242–247, Mar. 1975.

[21] T. P. Kelsey, K. K. Saluja, and S. Y. Lee, "An Efficient Algorithm for Sequential Circuit Test Generation," *IEEE Trans. Computers*, vol. 42, no. 11, pp. 1361–1371, Nov. 1993.

[22] S. Kundu, "Pitfalls of Hierarchical Fault Simulation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 2, pp. 312–314, Feb. 2004.

[23] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Technical Report 12_93, Dept. of Electrical Eng., Virginia Polytechnic Institute and State University, 1993.

[24] J. Lee and J. H. Patel, "Hierarchical Test Generation under Architectural Level Functional Constraints," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1144–1151, Sept. 1996.

[25] A. Lioy, "Looking for Functional Fault Equivalence," *Proc. International Test Conf.*, Oct. 1991, pp. 858–863.

[26] A. Lioy, "Advanced Fault Collapsing," *IEEE Design & Test of Computers*, vol. 9, no. 1, pp. 64–71, Mar. 1992.

[27] A. Lioy, "On the Equivalence of Fanout-Point Faults," *IEEE Trans. Computers*, vol. 42, no. 3, pp. 268–271, Mar. 1993.

[28] E. J. McCluskey and F. W. Clegg, "Fault Equivalence in Combinational Logic Networks," *IEEE Trans. Computers*, vol. C-20, no. 11, pp. 1286–1293, Nov. 1971.

[29] A. Motohara, M. Murakami, M. Urano, Y. Masuda, and M. Sugano, "An approach to fast hierarchical fault simulation," *Proc.* 25th *Design Automation Conference*, 1988, pp. 698–703.

[30] M. Nadjarbashi, Z. Navabi, and M. R. Movahedin, "Line Oriented Structural Equivalence Fault Collapsing," *IEEE Workshop on Model and Test*, 2000.

[31] T. M. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," *Proc. European Design Automation Conference*, Feb. 1991, pp. 214–218.

[32] 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 Conf.*, Oct. 2002, pp. 391–397.

[33] W. A. Rogers and J. A. Abraham, "CHIEFS: A Concurrent, Hierarchical and Extensible Fault Simulator," *Proc. International Test Conf.*, March 2005.

[34] R. K. K. R. Sandireddy, "Hierarchical Fault Collapsing for Logic Circuits," Master's thesis, Auburn University, Auburn, AL, May 2005.

[35] R. K. K. R. Sandireddy and V. D. Agrawal, "Diagnostic and Detection Fault Collapsing for Multiple Output Circuits," *Proc. Design, Automation and Test in Europe Conf.*, March 2005, pp. 1014–1019.

[36] D. R. Schertz and G. Metze, "A New Representation for Faults in Combinational Digital Circuits," *IEEE Trans. Computers*, vol. C-21, no. 8, pp. 858–866, Aug. 1972.

[37] C. E. Stroud, "AUSIM: Auburn University SIMulator - Version 2.1," Technical report, Dept. of Electrical & Computer Engineering, Auburn University, March 12, 2004.

[38] K. To, "Fault Folding for Irredundant and Redundant Combinational Circuits," *IEEE Trans. Computers*, vol. C-22, no. 11, pp. 1008–1015, Nov. 1973.

[39] R. R. Tummala and E. J. Rymaszewski, editors, *Microelectronics Packaging Handbook*. New York: Van Nostrand Reinhold, 1989. Page 677.

[40] A. Vaaje, "Theorems for Fault Collapsing in Combinational Circuits," *Journal of Electronic Testing: Theory and Application*. To be published.

[41] A. Veneris, R. Chang, M. S. Abadir, and M. Amiri, "Fault Equivalence and Diagnostic Test Generation Using ATPG," *Proc. IEEE International Symposium on Circuits and Systems*, May 2004, pp. 221–224.

[42] E. C. Weening and H. G. Kerkhoff, "A New Hierarchical Approach to Test-Pattern Generation," *Proc.* 4th *IEEE International ASIC Conference and Exhibit*, Sept. 1991, pp. P6–1/1–4.

[43] H. C. Wittmann, B. H. Seiss, and K. J. Antreich, "Using Circuit Hierarchy for Fault Simulation in Combinational and Sequential Circuits," *Proc.* 4th *European Conference on Design Automation*, Feb. 1993, pp. 432–436.

[44] P. Zhongliang, "Hierarchical Test Generation using Neural Networks for Digital Circuits," *Proc. International Conference on Neural Networks and Signal Processing*, Dec. 2003, pp. 245–248.