Fault Equivalence Identification Using Redundancy Information and Static and Dynamic Extraction

M. Enamul Amyeen, W. Kent Fuchs, Irith Pomeranz
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285

Vamsi Boppana
Fujitsu Labs of America
Lawrence Expressway
Sunnyvale, CA 94086

Abstract

A procedure for identifying functionally equivalent faults and improving the performance of diagnostic test pattern generation is described in this paper. The procedure is based on evaluation of faulty functions in cones of dominator gates of fault pairs. This is enhanced by utilizing circuit redundancy information. Equivalence is proved without the previously required circuit transformations. Stem-branch equivalences for reconvergent stems and their branches are identified efficiently obviating the need to check for non-masking and multiple-path sensitization. Both static and dynamic techniques are developed to exploit relations among inputs of dominator cones. This reduces the simulation time required by the procedure and enables evaluation of larger cones than could be evaluated earlier. As a result, more equivalent fault pairs are identified. Experiments performed on ISCAS85 circuits and full scan ISCAS89 circuits are used to demonstrate the effectiveness of the proposed techniques.

1 Introduction

Diagnostic test generation is both difficult and slow in the presence of a large number of functionally equivalent faults [1, 2]. This is similar to the problem of generating detection oriented test patterns in the presence of undetectable faults. Some early diagnostic test pattern generators [3–5] have not attempted to prove fault equivalence while others have used a variety of techniques with different degrees of success. Structural analysis [6] requires circuit graph manipulation, and is useful for fanout free regions of the circuit [7]. Other techniques have targeted two-level logic networks [8] or have required exhaustive enumeration [9]. The learning and implication based technique [10] has worst case exponential complexity. DIATEST [1] is a diagnostic test generation procedure that uses a branch and bound search procedure. It explores all possibilities to generate a distinguishing test for a fault pair before concluding that the faults are equivalent. Local circuit transformations and symmetric circuit identification have also been used for proving fault equivalence [2]. The procedure of [2] modifies the circuit structure and requires test generation to identify equivalent fault. Recent work [11] uses implication and evaluation techniques for proving functional equivalence of fault pairs. The implication technique is based on implication of faulty values, and comparison of fault effects at common successor and dominator gates. Evaluation techniques are based on evaluation of faulty functions at common dominator gates. The functions (represented as truth tables) are computed over a set of intermediate circuit lines that determine the value of the dominator gate output. These intermediate circuit lines define the inputs of a cone with the dominator as its output. If two faulty functions are identical for all the cone input combinations that can be justified by primary input combinations, then the corresponding faults are equivalent. Although the method identifies a large number of equivalent faults, it is limited to faults that require evaluation of cones with small numbers of inputs. Thus, some equivalent fault pairs may not be identified.

In this paper, we develop techniques that will allow us to evaluate larger cones and identify additional functionally equivalent fault pairs compared to [11]. Both static and dynamic techniques are used to find relations among dominator cone inputs. Using these relations, we can remove from evaluation combinations of cone input values which are not justifiable by any primary input combination. This reduces simulation time (since certain cone input combinations do not need to be evaluated) and permits evaluation of larger cones when trying to prove equivalence of two faults. Another distinguishing aspect of this paper compared to [11] is the utilization of redundant fault information during evaluation of faulty functions at dominator gates. We derive conditions under which a fault activated and propagated to a dominator gate output by a cone input combination will not be detected and, therefore, will not be distinguished from another fault that is not propagated to that dominator gate. Consideration of such input combinations can be avoided.

This paper is organized as follows. In section 2, we present definitions and background material. In section 3, we describe fault equivalence identification using circuit redundancy information. Static and dynamic techniques to extract relations among cone inputs are introduced in section 4. Experimental results are included in section 5.

2 Preliminaries

In this section, we introduce notation and definitions, and review the background required to describe the proposed techniques for identifying equivalent faults. We consider α and β which are a pair of stuck-at faults in a circuit N, with L_α and L_β being their corresponding faulty lines. The circuits in the presence of faults α and β are N_α and N_β, respectively.

Definition 1 (Functional Equivalence(α,β)) *If the circuit has n primary inputs $x_1, x_2, \ldots, x_n$ and m primary outputs $PO_1, PO_2, \ldots, PO_m$, and the functions corresponding to the primary outputs are $f_1, f_2, \ldots, f_m$, then fault α is functionally equivalent to fault β if and only if*

$$f_{i\alpha}(x_1, x_2, \ldots, x_n) = f_{i\beta}(x_1, x_2, \ldots, x_n), \forall i = 1, \ldots, m$$

This research was supported in part by the Office of Naval Research under contract N00014-97-1-1013, and in part by the Defense Advanced Research Projects Agency (DARPA) under contract DABT 63-96-C-0069.

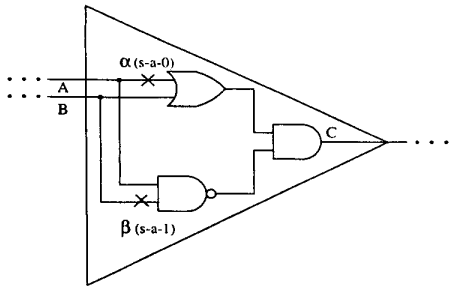

Figure 1: Functional Evaluation at a Dominator Gate

where $f_{i\alpha}(x_1, x_2, \ldots, x_n)$ and $f_{i\beta}(x_1, x_2, \ldots, x_n)$ are the respective faulty functions at primary output PO_i for faults α and β.

Due to the presence of a fault, certain nodes in a circuit are set to constant values. By removing these nodes we obtain a simplified circuit that realizes the same faulty function. Considering faults α and β, let the corresponding simplified circuits be $S(N_\alpha)$ and $S(N_\beta)$.

Definition 2 (Structural Equivalence(α,β)) *Faults α and β in logic circuit N are said to be structurally equivalent if and only if $S(N_\alpha) \equiv S(N_\beta)$ [6].*

Definition 3 (Dominator gate(l)) *A dominator gate of line l is a gate through which all the paths from l to any primary output pass [12–14].*

To show that two faults α and β are equivalent, we evaluate the faulty functions at the output of a common dominator gate of the faulty lines L_α and L_β by means of the dominator cone. The dominator cone is found by tracing the circuit back from the dominator gate output to the primary inputs. The trace is stopped as soon as we find intermediate nodes of the circuit that completely determine the faulty functions at the dominator gate output and are predecessors of L_α and L_β. Figure 1 is an example of a dominator cone. C is a common dominator gate of faults α and β. A and B are the inputs of the dominator cone. The good logic function at C in terms of A and B is $f(A, B) = A\overline{B} + \overline{A}B$, and the faulty functions are $f_\alpha(A, B) = \overline{A}B$ and $f_\beta(A, B) = \overline{A}B$.

In this paper, unless otherwise stated, by equivalence we mean functional equivalence. The following theorem is provided for completeness of discussion.

Theorem 1 *If the logic functions at the common dominator gate for faults α and β are identical when computed in terms of the inputs of the dominator cone, then faults α and β are functionally equivalent [11].*

We evaluate f_α and f_β at the common dominator gate by comparing their truth tables. The truth tables are written in terms of the inputs of the dominator cone. To show that faults α and β are equivalent, we need to show that any combination of the dominator cone inputs for which $f_\alpha \neq f_\beta$ cannot be justified by a primary input combination (i.e., there does not exist a primary input combination that results in this combination). In other words, if the logic values at the common dominator gate for fault α and fault β are different only for vectors $t_1, t_2, \ldots, t_m$ at the cone inputs, and none of $t_1, t_2, \ldots, t_m$ is justifiable, then faults α and β are functionally equivalent.

Diagnostic ATPG is typically applied when a fault detection test set is available. Thus, identification of equivalent faults starts from the fault pairs left indistinguished by a fault detection test set. We perform diagnostic fault simulation to obtain the indistinguished pairs that will be targeted by the proposed procedure.

3 Equivalence Identification Using Redundancy Information

This section describes techniques to identify functionally equivalent faults using information about redundant faults. Redundancy information is utilized during evaluation of faulty functions at a common dominator gate.

3.1 Local Circuit Transformations and Stem-branch Equivalence

Hartanto et al.[2] introduced local circuit transformations and stem-branch properties to identify equivalent faults utilizing circuit redundancy information. A local circuit transformation can help establish that fault pairs located at the terminals of a simple gate are functionally equivalent, even if they are not structurally equivalent, in the presence of redundant faults at other terminals of that gate. In addition to local circuit transformations, test generation is required to prove fault equivalence in [2]. Faults on a fanout stem and its branches can be functionally equivalent because of redundancies and reconvergence. To prove equivalence, the method of [2] requires analysis of self-masking and multiple-path sensitization.

The technique presented in this paper is different from the previous approach in that we use redundancy information in the context of functional evaluation at a common dominator gate. As a result, equivalent faults at the terminals of a gate are identified without performing local circuit transformations and without the associated test generation. Stem-branch equivalence for a reconvergent fanout stem and its branches is also identified without checking for non-masking and multiple-path sensitization.

3.2 Equivalence Identification through Functional Evaluation with Redundancy Information

In this section, we develop techniques to utilize redundant fault information during functional evaluation. As explained earlier, faulty functions are evaluated at a common dominator gate in terms of the cone inputs. If any cone input combination produces different output values at the dominator gate, a justification procedure is invoked to find a primary input combination that yields the cone input combination. If such a primary input combination is found, then the two faults may not be equivalent and we simply abort on that fault pair. Otherwise, additional cone input combinations (that result in different output values at the dominator gate) must be justified. We observe that if a cone input combination produces different faulty values at the dominator gate output, then exactly one of the faults is propagated to the dominator gate output. The following Lemma specifies when the fault effect of a fault propagated to a common dominator gate can be ignored, and justification of the cone input combination can be avoided using redundant fault information.

Lemma 1 *If a detected fault α and a redundant fault γ produce identical faulty values at a common dominator gate of α and γ for cone input combination $test_i$, then fault α will not be detected*

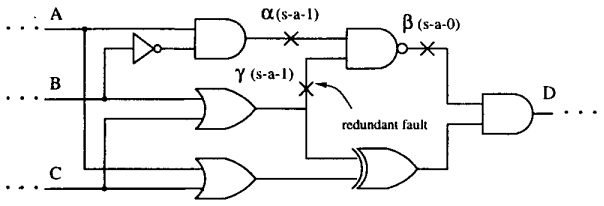

Figure 2: Equivalence of Terminal Fault Pair

Table 1: Simulation at Dominator Gate for Figure 2

A	B	C	$Good$	f_α	f_β	f_γ
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	0	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0

by any primary input combination that assigns $test_i$ to the cone inputs.

Proof Faults α and γ produce identical values at the common dominator gate for the vector $test_i$. Consequently, for all the primary input combinations that assign $test_i$ to the intermediate cone inputs, α and γ result in the same value at the common dominator gate. All the fault effects from α and γ go through their common dominator gate. Therefore, for all the primary input combinations that assign $test_i$, both faults generate the same values on all the primary outputs. Since γ is redundant, fault α is not detected by $test_i$.

Two examples of the application of Lemma 1 to the identification of equivalent faults are presented next. The first example shows how Lemma 1 can be utilized to prove equivalence of faults that are located at the terminals of a gate. The second example shows how it can be applied to prove equivalence of faults on a fanout stem and one of its branches.

Example 1 : In Figure 2, faults α and β share a common dominator gate D. Faulty functions f_α and f_β at the dominator gate D are evaluated in terms of the cone inputs A, B and C as shown in Table 1 (this cone was selected for the purposes of this example and may not be the one selected by our procedure). The faulty functions f_α and f_β have identical values at D for all the cone input combinations except for (1,0,0). For input combination (1,0,0), only the fault effect of β is propagated to the dominator gate. Since redundant fault γ produces the same output value 0 at the dominator gate for input combination (1,0,0), using Lemma 1, the mismatch between α and β can be ignored because the fault effect of β will not propagate to any primary output. Therefore, faults α and β are equivalent. Here, the local circuit transformation and the associated test generation step needed in [2] are not needed for proving equivalence.

Example 2 : Part of the circuit $s35932f$ is shown in Figure 3. We use it to illustrate the application of Lemma 1 to prove equivalence of faults on a stem and one of its branches.. Stem fault α and branch fault β are evaluated at the dominator gate $G11431$ as shown in Table 2. Faults γ and δ are redundant faults at the fanout branches of stem $G10935$. According to Table 2, for input combination (1,0,0) and (1,0,1), faults α and β result in different values

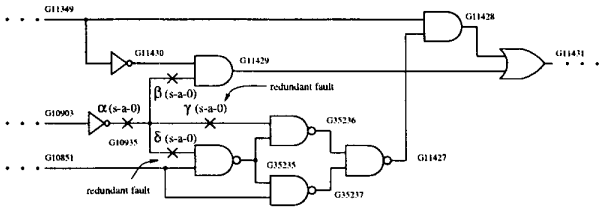

Figure 3: Equivalence at Reconvergent Stem

Table 2: Simulation at Dominator Gate for Figure 3

$G11349$	$G10903$	$G10851$	$Good$	f_α	f_β	f_γ	f_δ
0	0	0	1	0	0	1	1
0	0	1	1	0	0	1	1
0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0
1	0	0	1	0	1	0	1
1	0	1	0	1	0	0	1
1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1

at the dominator gate. In both cases, only the fault effect of α is propagated to the common dominator gate $G11431$. Since redundant fault γ produces an identical faulty value of 0 for combination (1,0,0), and redundant fault δ results in an identical faulty value of 1 for combination (1,0,1), using Lemma 1, the fault effects of α for these combinations can be ignored. For all other input combinations, faults α and β produce identical values, hence they are equivalent.

4 Static and Dynamic Evaluation

In this section, static and dynamic techniques are described that find relations among inputs of the common dominator cone in order to accelerate functional evaluation. The relations are used for identifying cone input combinations that cannot be justified. This will eliminate the need to simulate and justify them, and will reduce the time spent on evaluation.

4.1 Motivational Example

The fault pair considered here to illustrate the need for finding relations among cone inputs is from the ISCAS85 benchmark circuit $c7552$. The common dominator cone has 10 inputs denoted IN[0], IN[1], ..., IN[9]. The structural relations among these inputs are shown in Figure 4. Inputs IN[2] and IN[3] are structurally related in such a way that a value 1 at IN[2] implies a value 0 at IN[3]. Therefore, the combination $(1,1)$ at (IN[2],IN[3]) cannot be justified by any primary input combination. Hence, any vector at the dominator cone inputs with $(1,1)$ at (IN[2],IN[3]) is not justifiable and can be skipped. The inputs IN[1] and IN[4] are identical and combinations $(0,1)$ or $(1,0)$ at (IN[1],IN[4]) cannot be justified. Similarly, combinations $(0,1)$ and $(1,0)$ at identical inputs (IN[7],IN[9]) are not justifiable. The input IN[1] is the predecessor of IN[0] and vectors containing $(1,0)$ at (IN[0],IN[1]) or (IN[0],IN[4]) can be skipped. Out of 1024 test vectors at the dominator cone, 880 vectors do not need evaluation because they cannot be justified based on structural relations among the cone inputs. The reduced computation time allows functional evaluation to be extended to larger cone sizes, and leads to the identification of more equivalent pairs.

4.2 Static Structural Property Extraction

Motivated by the example above, two relations among the dominator cone inputs are derived. First, predecessor-successor rela-

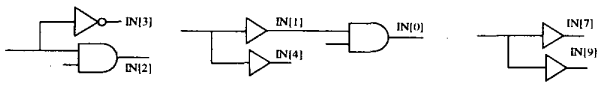

Figure 4: Structural Relations Between Dominator Cone Inputs

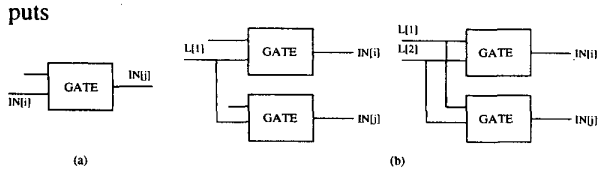

Figure 5: Static Analysis (a) Predecessor Successor Relations (b) Successor Successor Relations

tions are determined, i.e., whether one dominator cone input is a predecessor or successor of another input as shown in Figure 5(a). If the predecessor input IN[i] has a controlling value, c, and the inversion of the gate is k, then the combination (c,$\overline{c \oplus k}$) on (IN[i],IN[j]) cannot be justified.

The second relation between cone inputs is the successor-successor relation, which occurs if two dominator cone inputs are successors of other lines. In Figure 5(b), IN[i] and IN[j] are successors of line L[1] (or lines L[1] and L[2]). If $value_i$ at IN[i] implies $value_j$ at IN[j] then the combination ($value_i$,$\overline{value_j}$) at (IN[i],IN[j]) cannot be justified.

We search for structural properties of dominator cone inputs statically before starting the evaluation process. Extraction and utilization of input relations is incorporated into the evaluation procedure. During functional evaluation, a check is made to determine whether the current vector is unjustifiable based on the structural properties and can be skipped.

The relations obtained from the structural technique can also be identified using static learning techniques [15]. The advantage of the structural technique is that it processes circuit information locally, whereas the implications required in static learning [15] may involve larger areas of the circuit.

4.3 Dynamic Justification and Property Extraction

Consider a vector at the dominator cone inputs that produces different faulty values at the dominator gate output. Suppose that the vector cannot be shown to be unjustifiable based on the information derived during static analysis of the cone inputs. In this case, we use the multiple backtrace procedure of FAN [13] to find out if the vector is justifiable or not. The goal of dynamic justification described in this subsection is to extract as much information as possible from a vector that was already proved to be unjustifiable. Specially, our goal is to extract new relations among cone inputs based on the unjustifiable vector.

When a cone input vector produces different faulty values at the common dominator gate and cannot be justified, the vector is analyzed to find a smaller set of inputs that are unjustifiable (if such a set exists). For this purpose, we divide the inputs into different groups and use binary search. Figure 6 shows the input partition, which is used as follows. First, block 1 and block 2 are set to the same values as the unjustified vector and all the other inputs are set to the don't care value ('X'). The justification procedure is then applied to this vector. If this vector is not justifiable, then we

Figure 6: Partitioning of a Test Vector

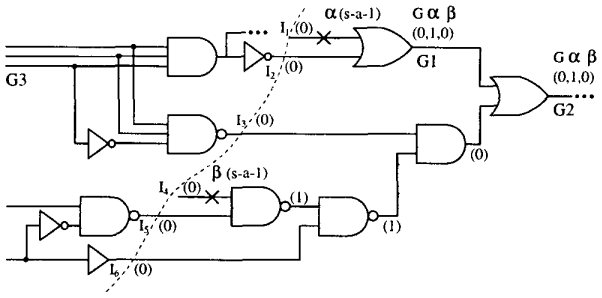

Figure 7: Dynamic Justification Analysis

know that unique unjustifiable inputs reside in block 1 and block 2. These blocks are further partitioned into two groups to search for an even smaller unjustifiable input set. If the vector defined over blocks 1 and 2 is justified, then the other two-block combinations, blocks (3,4), blocks (1,3), blocks (1,4), blocks (2,3) and blocks (2,4) are processed in the same manner. When the 2-way partitioning fails, 3-way partitioning is tried to find unjustifiable input combinations in blocks (1,2,3), blocks (1,2,4), blocks (1,3,4) and blocks (2,3,4). This technique can be extended to higher levels of partitioning, but in most cases we were able to extract a smaller unjustifiable input set with 2-way and 3-way partitions. In the worst case, after 20 partitions, the smallest unjustified input set obtained so far is returned or we abort the process.

Figure 7 illustrates the dynamic approach to extracting relations among the dominator cone inputs. The cone inputs for faults α and β with respect to common dominator gate $G2$ are lines I_1, I_2, I_3, I_4, I_5 and I_6. Let the fault free function at the dominator gate be f, and let the faulty functions be f_α and f_β. During evaluation, the faulty functions are computed for vector (0,0,0,0,0,0). In Figure 7, we show the values obtained as tuples (G,α,β) where G denotes the value of the good circuit, α denotes the value due to fault α, and β denotes the value due to fault β. The evaluation at gate $G2$ shows that only fault α is propagated to the output of the dominator gate. We use the multiple backtrace procedure of the FAN [13] algorithm for justification. The procedure shows that the vector (0,0,0,0,0,0) is not justifiable.

In order to find a smaller set of unjustifiable inputs, we partition the inputs into four blocks, $[I_1]$, $[I_2, I_3]$, $[I_4]$ and $[I_5, I_6]$. First, block 1 and block 2, i.e., $[I_1]$ and $[I_2, I_3]$ are set to the same values as the unjustified vector and all the other inputs are set to 'X'. The justification procedure is then called with vector $(I_6, I_5, I_4, I_3, I_2, I_1) = (X, X, X, 0, 0, 0)$. This vector is unjustifiable and we further partition the block $[I_1, I_2, I_3]$ into $[I_1]$ and $[I_2, I_3]$ and repeat the justification step with $(X, X, X, X, X, 0)$. The vector $(X, X, X, X, X, 0)$ is justifiable and justification is tried for $(X, X, X, 0, 0, X)$. This vector is unjustifiable, and the smaller unjustifiable input set $(I_3, I_2) = (0, 0)$ is found. We cannot find smaller unjustifiable sets, therefore, this relation is stored, and simulation for subsequent vectors that match $(X, X, X, 0, 0, X)$ will be skipped. The evaluation phase again

produces a mismatch for the vector $(1,0,0,1,0,0)$. If we decide to perform dynamic justification, we will obtain unjustifiable inputs $(I_6, I_5) = (1,0)$. Storing this relation, we will skip the remaining vectors that match $(1,0,X,X,X,X)$.

When the number of remaining cone input vectors that need to be checked is low, the overhead of dynamic justification exceeds the gain obtained from the extracted relations. To avoid this and to obtain maximum benefit from dynamic extraction, we use several guidelines. The dynamic extraction procedure is applied only when the cone size is 12 or greater. The larger the cone size, the larger the number of simulations and justifications that can be skipped. The gain obtained from subsequent calls to dynamic extraction decreases. Therefore, we set a limit of at most five calls to dynamic extraction for each fault pair. Finally, we avoid dynamic extraction when the unjustified vector requires no or very few backtracks.

5 Experimental Results

All the experiments reported in this section were executed on a SUN Ultrasparc 2 workstation with 512 MB of memory. All the ISCAS85 benchmarks and full scan versions of the larger ISCAS89 benchmarks were utilized in the experiments.

Fault Pairs Left Indistinguished after Applying a Fault Detection Test Set

Diagnostic ATPG is typically applied only after a fault detection test set is available. Thus, identification of equivalent fault pairs starts from the fault pairs left indistinguished by a fault detection test set. We used the following process to find indistinguished fault pairs. First, redundant faults were removed from the fault list, since all the redundant faults belong to the same equivalence class and bias the number of indistinguished pairs. Then, diagnostic fault simulation was performed using RAPSIM [16] to obtain the indistinguished pairs. The test vectors used are complete detection test sets [17, 18]. Column 2 of Table 3 shows the number of fault pairs that remain indistinguished after applying a fault detection test set.

Following diagnostic fault simulation, we identified equivalent fault pairs using the implication technique from [11]. The number of equivalent fault pairs identified by implications are shown in column 3 of Table 3. Column 4 shows the run time for the implication procedure, and indicates that a fraction of a second is required for proving equivalence by implication. For example, it takes less than 1 second to identify 1179 equivalent pairs in circuit $s35932f$.

Results with Static and Dynamic Extraction

Next, we use functional evaluation at common dominator gates to identify additional equivalent fault pairs. In this experiment, we do not use redundant fault information. We only consider fault pairs for which we can find a common dominator gate, and for which the dominator cone has at most 17 inputs. Results are provided in Table 3. Columns 5 and 6 show results of proving equivalence without static or dynamic extraction. This is the evaluation method proposed in [11]. The total number of fault pairs proved to be equivalent is given in column 5. The next column shows the total time for identifying equivalent fault pairs. Results using the static and dynamic analysis techniques proposed here are given in columns 7 and 8. Using static and dynamic analysis, we obtain an increase in the total number of faults identified as equivalent for all the benchmarks except $c499$, $c880$, $c1355$ and $s35932f$. Among these all the equivalent pairs of $c499$, $c880$ and $c1355$ are already identified. The improvement obtained by static and dynamic analysis is due to the extraction of relations among cone inputs which enable us to evaluate cones with 17 inputs within reasonable run times compared to 10 inputs in the previous case [11]. For example, in circuit $s15850f$, the total number of equivalent pairs identified increases from 2268 to 2515, and time increases from 53 seconds to 138 seconds. Pairs proven equivalent through static and dynamic functional evaluation take less than 140 seconds for all the circuits. For example, a total of 8898 pairs in $s35932f$ are proved equivalent in less than 4 seconds.

To further demonstrate the advantages of using static and dynamic extraction, we considered circuit $s15850f$ using cones with 17 inputs under the previous evaluation method [11], and compared the results to the results obtained using static and dynamic extraction. The previous method [11] identified 2515 equivalent fault pairs and took 367.6 seconds, while evaluation with static and dynamic analysis identified the same 2515 equivalent fault pairs but took only 138.1 seconds. The diagnostic test generator DIATEST [1] took 318.2 seconds for the same fault pairs.

Results Using Redundancy Information

Results of the identification of equivalent fault pairs using redundancy information are presented in the last two columns of Table 3. Here we consider pairs left after applying the implication technique [11] and the evaluation technique with static and dynamic analysis. Column 9 shows the total number of equivalent fault pairs identified using redundancy information. The last column shows the total CPU time to prove these equivalences. For $s35932f$, we identify 2048 equivalent pairs in approximately 402 seconds. All the equivalent pairs of $c499$, $c880$, $c1355$ and $c5315$ are already identified by the evaluation technique (with static and dynamic analysis) and the implication technique [11].

The number of equivalent fault pairs identified by using all the techniques proposed here are reported in Table 4 and compared with several other results. Column 2 shows the number of equivalent pairs obtained using the diagnostic test generation procedure DIATEST [1]. Column 3 gives the number of pairs obtained previously [11] using implication and evaluation without utilizing redundancy information or static and dynamic extraction. The next column shows the total number of equivalent pairs found using the implication technique [11], evaluation with static and dynamic extraction and the redundancy based techniques proposed here. Columns 5 and 6 show the number of equivalent pairs that were not proved to be equivalent. Using the techniques proposed here on top of [11], we achieve a considerable increase in the number of equivalent fault pairs identified. It is observed that only a small number of equivalent pairs remain unidentified and over 95.7% of all the equivalent pairs are identified. Out of 12893 pairs, 12125 pairs of $s35932f$ are proven equivalent.

The remaining fault pairs cannot be identified as equivalent due to the following reasons. Some fault pairs cannot be identified due to the large dominator cone size that needs to be considered. A fraction of these pairs are far apart in the circuit and do not have a single dominator gate, or the cone has more than 17 inputs. A few faults exceeded the backtrack limit during justification, or primary input combinations could be found to justify the necessary cone

Table 3: Equivalence Identification Using Static and Dynamic Analysis and Redundancy Information

Circuit	Indist. Pairs	Impli. Pairs	Time (sec)	w/o Static & Dynamic Anal		w/ Static & Dynamic Anal		w/ Redundancy	
				Proven Pairs [11]	Time (sec)[11]	Proven Pairs	Time (sec)	Proven Pairs	Time (sec)
c432	83	9	0.00	0	0.00	1	67.07	0	0.00
c499	21	0	0.00	12	0.30	12	0.02	0	0.00
c880	61	1	0.00	54	0.03	54	0.02	0	0.00
c1355	775	72	0.00	668	0.60	668	0.27	0	0.00
c1908	346	59	0.00	218	0.78	223	1.35	0	0.00
c2670	616	14	0.00	297	0.27	358	6.52	95	0.06
c3540	574	73	0.00	398	5.28	409	2.09	31	0.06
c5315	537	8	0.00	432	3.38	439	1.32	0	0.00
c6288	1327	0	0.00	930	0.45	945	0.35	16	0.00
c7552	1210	7	0.00	864	1.23	944	1.22	137	0.10
s5378f	550	8	0.00	479	2.72	489	4.53	4	0.07
s9234f	1370	41	0.00	957	4.63	1100	61.49	32	3.54
s13207f	2064	64	0.03	1638	9.47	1758	84.8	30	1.32
s15850f	2842	49	0.03	2268	52.83	2515	138.12	111	46.43
s35932f	12899	1179	0.98	8898	5.07	8898	3.82	2048	401.57
s38417f	4401	182	0.17	3076	8.04	3121	14.74	4	0.96
s38584f	2797	68	0.05	2261	7.73	2435	23.15	89	5.55

input vectors (this means that faults are indistinguishable because they cannot be propagated to different outputs).

Table 4: Overall Equivalence Identification

Circuit	Equiv. Pairs	Proven Pairs [11]	Proven Pairs	Remaining Pairs [11]	Remaining Pairs
c432	13	9	10	4	3
c499	12	12	12	0	0
c880	55	55	55	0	0
c1355	740	740	740	0	0
c1908	295	277	282	18	13
c2670	468	311	467	157	1
c3540	531	471	513	60	18
c5315	447	440	447	7	0
c6288	1013	930	961	83	52
c7552	1118	871	1088	247	30
s5378f	523	487	501	36	22
s9234f	1229	998	1173	231	56
s13207f	2043	1702	1852	341	191
s15850f	2789	2317	2675	472	113
s35932f	12893	10077	12125	2816	768
s38417f	3361	3258	3307	103	54
s38584f	2696	2329	2592	367	104

Table 5 shows the execution time compared to DIATEST [1]. The run time shown here for DIATEST is obtained on the same SUN workstation we use for identifying equivalent fault pairs, and DIATEST is applied only to the faults identified as equivalent by the procedure of this paper. The results show that the presented technique is significantly faster in identifying equivalent fault pairs. For $s35932f$, the approach of this paper takes 7 minutes whereas DIATEST takes over two hours and 22 minutes for the same pairs.

Next, we demonstrate the improvement in the total diagnostic test generation time due to the proposed procedure. For this purpose, the equivalent fault identification tool is used at the front end of the DIATEST diagnostic test pattern generator [1]. The results are shown in Table 6. Column 2 in Table 6 gives the number of indistinguished pairs after applying the complete fault detection test set. Column 3 shows the total number of pairs proven equivalent by the technique of this paper. Time t_p is the time required for findiing equivalent fault pairs. Time t_r is the time required by DIATEST for the remaining pairs and for diagnostic test generation. The total time for DATPG using the new approach is $t_p + t_r$. The total time (without using our tool) required by DIATEST [1] is given in the next column. The last two columns show the savings obtained in DATPG time here and in the previous case [11]. For

Table 5: Comparison with DIATEST in Identifying Equivalences

Circuit	Proven Pairs	DIATEST (sec)	Ours (sec)
c432	10	71.4	67.1
c499	12	0.8	0.0
c880	55	0.4	0.0
c1355	740	12.1	0.3
c1908	282	5.0	1.4
c2670	467	317.6	6.6
c3540	513	17.9	2.2
c5315	447	37.8	1.3
c6288	961	42.7	0.4
c7552	1088	107.8	1.3
s5378f	501	15.7	4.6
s9234f	1173	333.7	65.0
s13207f	1852	165.8	86.2
s15850f	2675	345.3	184.6
s35932f	12125	8569.8	406.4
s38417f	3307	1117.0	15.9
s38584f	2592	3278.8	28.8

$c2670$, savings of 2.1% reported previously [11] are increased to 96.8%. For $s35932$, the savings increase from 46.0% to 79.9%. In a few circuits, we see a decrease in savings in DATPG time compared to the previous case [11]. For these circuits, the savings are lower because of the additional time it takes to identify additional fault pairs, but still the savings are significant compared to DIATEST. Savings of up to 89% are obtained for the largest benchmarks and the average for all the circuits is 62% compared to the previous [11] average savings of 48%.

6 Summary

Techniques were introduced to identify functional equivalence utilizing redundancy information during faulty function evaluation at common dominator gates. We derived conditions under which the effect of a fault propagated to the dominator gate could be ignored and justification for the cone input vector could be avoided. Static and dynamic techniques were developed to find relations among dominator cone inputs. Precomputed structural relations and extracted justification properties were used to reduce simulation time. It was shown that utilizing these techniques, we were able to rapidly identify most of the equivalent faults in benchmark circuits and obtain significant savings in diagnostic test pattern generation time.

The procedure described here can be improved by considering forward justification during functional evaluation, i.e., by show-

Table 6: Savings in Total DATPG Time

Circuit	Indist. Pairs	Proven Pairs	t_p (sec)	t_r (sec)	$t_p + t_r$ (sec)	Total(sec) DIATEST	datpg savings	datpg [11] savings
c432	83	10	67.1	274.7	341.8	346.1	1.3%	0.03%
c499	21	12	0.0	0.2	0.2	1.0	80.0%	40.0%
c880	61	55	0.0	0.1	0.1	0.5	80.0%	60.0%
c1355	775	740	0.3	1.0	1.3	13.1	90.1%	89.3%
c1908	346	284	1.4	3.4	4.8	8.4	42.9%	45.2%
c2670	616	467	6.6	3.7	10.3	321.3	96.8%	2.1%
c3540	574	513	2.2	4.5	6.7	22.4	70.1%	42.0%
c5315	537	447	1.3	2.9	4.2	40.7	89.7%	79.1%
c6288	1327	961	0.4	14.7	15.1	57.4	73.7%	70.2%
c7552	1210	1088	1.3	88.4	89.7	196.2	54.3%	31.5%
s5378f	550	509	4.6	4.9	9.5	20.6	53.9%	64.6%
s9234f	1370	1182	65.0	1783.8	1848.8	2117.4	12.7%	6.3%
s13207f	2064	1857	86.2	85.2	168.7	248.2	32.0%	49.7%
s15850f	2842	2677	184.6	131.0	315.6	476.2	33.7%	45.3%
s35932f	12899	12893	406.4	1652.9	2059.3	10222.7	79.9%	46.0%
s38417f	4401	3312	15.9	333.7	349.6	1450.7	75.9%	70.2%
s38584f	2797	2620	28.8	353.7	382.5	3632.5	89.5%	77.7%

ing that the mismatch produced by two faults at the dominator gate cannot be propagated to any of the primary outputs. In the current implementation, the presence of redundant faults allows us to indirectly derive information about forward propagation. In the absence of redundant faults, we use backward justification to show that a cone input vector which produces different values at the dominator gate cannot be justified by any primary input combination. It may also happen that backward justification alone is unable to resolve the discrepancy at the dominator gate, i.e., the cone input vector which produces different values for two faults is justifiable by the primary inputs. In such a case, we will need forward processing to prove that the mismatch does not propagate to any primary outputs. From our experiments with benchmark circuits, we found that this occurs only in a small number of cases.

References

[1] T. Grüning, U. Mahlstedt, and H. Koopmeiners, "DIATEST: A fast diagnostic test pattern generator for combinational circuits," *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp. 194–197, Nov. 1991.

[2] I. Hartanto, V. Boppana, and W. K. Fuchs, "Diagnostic fault equivalence identification using redundancy information & structural analysis," *Proc. IEEE Intl. Test Conf.*, pp. 294–302, Oct. 1996.

[3] J. Savir and J. P. Roth, "Testing for, and distinguishing between failures," *Proc. IEEE Intl. Fault Tolerant Computing Symp.*, pp. 165–172, June 1982.

[4] P. Camurati, A. Lioy, P. Prinetto, and M. Sonza Reorda, "Diagnostic oriented test pattern generation," *Proc. European Design Automation Conf.*, pp. 470–474, Mar. 1990.

[5] P. Camurati, D. Medina, P. Prinetto, and M. Sonza Reorda, "A diagnostic test pattern generation algorithm," *Proc. IEEE Intl. Test Conf.*, pp. 52–58, Sept. 1990.

[6] 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.

[7] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital System Testing and Testable Design*, New York, NY: Computer Science Press, 1990.

[8] 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.

[9] B. K. Roy, "Diagnosis and fault equivalences in combinational circuits," *IEEE Trans. Computers*, vol. C-23, no. 9, pp. 955–963, Sept. 1974.

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

[11] M. E. Amyeen, W. K. Fuchs, I. Pomeranz, and V. Boppana, "Implication and evaluation techniques for proving fault equivalence," *Proc. IEEE VLSI Test Symp.*, pp. 201–207, Apr. 1999.

[12] R. E. Tarjan, "Finding Dominators in Directed Graphs," *SIAM Journal of Computing*, pp. 62–89, Mar. 1974.

[13] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. Computers*, vol. C-32, pp. 1137–1144, Dec. 1983.

[14] T. Kirkland and M. R. Mercer, "A Topological Search Algorithm for ATPG," *Proc. IEEE/ACM Design Automation Conf.*, pp. 502–508, June 1987.

[15] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Trans. Computer-Aided Design*, vol. 7, no. 1, pp. 126–137, Jan. 1988.

[16] S. Venkataraman, I. Hartanto, W. K. Fuchs, E. M. Rudnick, S. Chakravarty, and J. H. Patel, "Rapid diagnostic fault simulation of stuck-at faults in sequential circuits using compact lists," *Proc. IEEE/ACM Design Automation Conf.*, pp. 133–138, June 1995.

[17] S. M. Reddy, I. Pomeranz, and S. Kajihara, "Compact Test Sets for High Defect Coverage," *IEEE Trans. Computer-Aided Design*, vol. 16, no. 8, pp. 923–930, Aug. 1997.

[18] I. Hamzaoglu and J. H. Patel, "New techniques for deterministic test pattern generation," *Proc. IEEE VLSI Test Symp.*, pp. 446–452, Apr. 1998.