Efficient Automatic Diagnosis of Digital Circuits

Heh-Tyan Liaw, Jia-Horng Tsaih, & Chen-Shang Lin

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC.

Abstract — Diagnosis is a process to locate and correct design errors of an erroneous system. The problem of automatic diagnosis of digital circuits with efficiency is studied in this paper. Two improvements over the method of [1] are developed to enhance the efficiency of diagnosis. Specifically, the dominance relation in circuit topology is utilized to reduce the search space of possibly correctable gates. In our experiment, the search space is reduced to about 1/2. And a novel divide-and-conquer technique to determine the correct gate function is proposed.

I. Introduction

When a circuit design under verification is determined to be erroneous, a procedure must be applied to locate and correct the design errors. This diagnosis process often takes a significant portion of design time. Diagnosis systems have been discussed in recent years. VERIFIER[2] isolates the errors into an area which might be as large as 35% of the fan-in cones of erroeous outputs. Reiter[3] tries to determine the set of possibly incorrect gates by enumerating input patterns which make the errors observable. None of them support the automatic correction of errors.

Under the single error assumption, Madre et al.[1] invoke boolean equation theory to find possibly correctable gates, and then try to solve the gate function for each of these gates. The solution, if exists, is the desired gate function to correct the circuit error. These two phases can be viewed as a screen-then-correct approach to achieve efficiency for error correction. However, their method in solving the gate function still contains high complexity when the number of primary inputs is large.

Two improvements over the method of [1] are developed to enhance the efficiency of diagnosis. Specifically, the dominance relation in circuit topology is utilized to reduce the search space of possibly correctable gates. In our experiment, the search space is reduced to about 1/2. And a novel divide-and-conquer technique to determine the correct gate function is proposed.

In the next section, the basic principle of diagnosis for the single-error case is introduced. Then the dominance relation is developed to reduce the search space of possibly correctable gates. In section IV, a divide-and-conquer approach to efficiently modify the gate in order to correct the circuit function is presented. And finally a conclusion is given.

II. Automatic Diagnosis Principle

In this section, the diagnosis problem of the digital circuits will be investigated. Since a synchronous sequential circuit can be characterized by its next-state function and output function[4], it is sufficient to develop the technique for the combinational circuits.

Consider a gate-level combinational circuit N with n inputs, $x=(x_1,x_2,...,x_n)$, and m outputs, $y=(y_1,y_2,...,y_m)$. In the verification, the function $y=F(x)$ derived from the circuit is compared with the given specification function $S(x)$. For an error-free designed circuit, $F(x)=S(x)$. And if there are some j's such that $F_j(x)\neq S_j(x)$, then the designed circuit is in error.

The diagnosis problem is that given an erroneous circuit N and its associated correct $S(x)$, locate and modify some gates in N such that the function $F'(x)$ of the modified circuit N' is equal to $S(x)$. The problem can be solved if the number of gates to be modified is small. We will consider the case of correction by single-gate modification. The single-gate modification case will be simply called single-error case with the understanding that the modified gate may not be original erroneous gate since there are more than one ways to correct a circuit. We further assume that all gates are of single output since the results can be easily extended to the more general cases.

The solving of the diagnosis problem can be carried out in two phases as suggested in [1]. The first phase is to identify those gates in N, the output of which, when injected an appropriate signal, produces the correct values at the primary outputs for all possible inputs x. These gates are called s-correctable gates. Any gate whose modification can correct the error in N must be in the set of s-correctable gates. Then in the second phase, the circuit can be corrected by exhaustively modifying the s-correctable gates. When solutions exist, any one of them is sufficient. The gate which can be a solution is named f-correctable. On the other hand, if there is no solution, then the erroneous circuit can not be corrected by modifying one single gate. In this case, one may try modifying more than one gates simultaneously. More formal definitions can be given as follows.

DEFINITION:

Let the output of a gate G in N be replaced by a new variable z, and the modified circuit-output function be $F(x,z)$. If there exists a boolean function $z=z(x)$ such that $F(x,z(x))=S(x)$ then G is s-correctable. Suppose that $v_1,...,v_p$ are the inputs of gate G, and $f_i(x)$, $i=1,...,p$, are the corresponding functions of v_i with respect to primary input vector x. If there exists a function

CH2924-9/90/0000/0464$01.00 © 1990 IEEE

$h=h(v_1,...,v_p)$ such that $F(x,h(f_1(x),...,f_p(x)))=S(x)$, then gate G is f-correctable.

The s-correctability can be tested by symbolic simulation such as that in [5], and the following classical theorem by Schröder.

PROPOSITION 1 [6]:

For a boolean expression F(x,z), where x in B^n , z in B, and B={0,1}, let L(x)=F(x,0), H(x)=F(x,1). Then the necessary and sufficient condition for the existence of function z=z(x) in

$$F(x,z(x))=0 \text{ for all } x$$

is

$$L(x)H(x)=0 \text{ for all } x.$$

And when the above condition is satisfied, the complete solution can be expressed as

$$z = L + H'A$$

where A=A(x) is an arbitrary boolean function.

Note that the "resolution procedure" presented and proved in [1] is the duality of the above theorem.

For multiple-output circuits, all primary outputs must be considered simultaneously. Let F(x,z) and S(x) are the derived function and the specification, respectively. For j=1,2,...,m, let $E_j(x,z) = F_j(x,z)$ [xor] $S_j(x)$, and let function $E=E_1+E_2+...+E_m$. Then

$$E(x,0)=E_1(x,0)+E_2(x,0)+...+E_m(x,0),$$
$$E(x,1)=E_1(x,1)+E_2(x,1)+...+E_m(x,1).$$

Based on Proposition 1, it can be easily shown that the s-correctability can be tested by the following proposition.

PROPOSITION 2:

The following statements are equivalent:

(a) There is a boolean function z=z(x) such that

$$E_j(x,z(x))=0 \text{ for all x and j=1,2,...m,}$$

(b) There is a boolean function z=z(x) such that

$$E(x,z(x))=0 \text{ for all x,}$$

(c) $[L_1(x)+L_2(x)+...+L_m(x)]\,[H_1(x)+H_2(x)+...+H_m(x)] = 0$ for all x, where $L_j(x)=E_j(x,0)$ and $H_j(x)=E_j(x,1)$,

(d) For all i,j=1,2,..,m, $L_i(x)H_j(x)=0$ for all x.

And when the above condition is satisfied, the complete solution can be expressed as

$$z(x) = (L_1+L_2+...+L_m) + (H_1+H_2+...+H_m)'\,C,$$

where C=C(x) is an arbitrary function.

It can be seen that the efficiency of checking the s-correctability of a gate depends on the size of the functions L and H. In our experience, significant speedup can be achieved by employing the necessary condition $L_j(x)H_j(x)=0$ as a filter to screen out those uncorrectable gates.

Since the condition in Proposition 2 must be checked for each gate in the circuit, it is crucial to reduce the search space. The reduction of search space will be discussed in the next section.

III. Reduction of Search Space

It is clear that, under the single-error assumption, only the gates in the intersection of fan-in cones of erroneous outputs are to be searched for the s-correctable gates. The search space of s-correctable gates can be further reduced by taking the dominance relation into consideration.

A combinational circuit can also be viewed as a directed acyclic graph in which each vertex corresponds to a logic gate and a directed edge (U,V) corresponds to the connection from gate U to gate V. A vertex V is said to dominate a vertex U, or V is a dominator of U, if every directed path from U to any primary output must go through V. A vertex V is said to be the immediate dominator of vertex U, denoted by V=idom(U), if V is a dominator of U and V is in turn dominated by every other dominators of U. Note that our definition of dominance relation is essentially the same as that of [7]. The dominance relation of the circuit can be represented by a dominance forest. In each tree of the dominance forest, the parent of each vertex V is idom(V), and all the dominators of a vertex V are the ancestors of V in the dominator tree.

In a circuit, it can be seen that if gate V dominates gate U, then U is s-correctable only if V is s-correctable. The following proposition allows us to locate s-correctable gates (vertices) by examining basically only the roots of dominator trees which include the degenerate trees with only one vertex.

PROPOSITION 3 :

In a circuit, a gate V is s-correctable only if idom(V) is s-correctable.

In the search of s-correctable gates, whenever a gate V is found to be not s-correctable, the entire subtree rooted at V can be safely skipped. The benifit of utilizing dominance relation can be estimated by the ratio of the dominance tree number to the total gate number in a circuit. These ratios for several circuits are listed in the last column of Table 1. Here Add32 is a gate-level 32-bit adder, ALU32 is constructed by cascading eight 74181 4-bit ALUs, and others are ISCAS benchmarks. Their ratios are from 16% to 57%. Also note that the overhead of finding dominators is negligible, less than 0.1s on SPARC server 390, for all circuits . In fact, it can be completed in a time linearly proportional to the size of the circuit[8].

Our diagnosis algorithm based on Proposition 3 is shown in Fig.1. The experimental data of the algorithm is shown in Table 2. In the experiment, we equally divided the gates in each circuit into 5 groups according to a topological order, and then randomly selected a gate in each group to inject an error. In each circuit, the error in case a is the one closest to primary inputs, while the error in case e is closest to the primary outputs.

In the diagnosis algorithm, the phase of finding the f-correctable gates (lines 16--19) will be described in the next section.

IV. Correction

From Proposition 2, a gate is s-correctable if and only if LH=0 and the correct value at the gate should be

$$z = L + H'C,$$

where C(x) is an arbitrary parametric function of primary input vector x. Let $v_1,...,v_p$ are inputs of the s-correctable gate, then the gate is f-correctable if and only if there exists a function of p inputs $h=h(v_1,...,v_p)$ such that

$$h(f_1(x),...,f_p(x))=L(x)+H'(x)C(x) \text{ for all x,} \quad (1)$$

where $f_1(x),...,f_p(x)$ are the functions of $v_1,...,v_p$, respectively. If the function solution h exists, then the circuit can be corrected by replacing the original gate with one of function h.

It appears that the direct application of Eq.1 would leads us to the solution function h for $h(f_1,...,f_p)=f$ where $f_1,f_2,...,f_p$ are the

functions of the gate fan-ins and f is obtained from Eq.1. However, to test the existence of h in this way, it is necessary to try all possible 2^{2^n} parametric function C(x) in the worst case where n is the number of primary inputs. It is not clear how this difficulty can be avoided[1].

The difficulty of the parametric function C(x) in Eq.1 can be circumvented by noting that solving Eq.1 is equivalent to checking the original condition:

$$E(x, h(f_1(x),...,f_p(x)))=0 \text{ for all } x,$$

or

$$h'(f_1(x),...,f_p(x))L(x)+h(f_1(x),...,f_p(x))H(x)=0 \text{ for all } x. \quad (2)$$

Note that in Equation 2, all functions except h are known. However, this brute-force approach is exponential in essence. In a typical circuit, say, p=3, n=32, the ordered binary decision diagram(OBDD) for $F=h(f_1,f_2,f_3)$ is of n-level and may possess 2^n-1=4,000,000,000 vertices in the worst case, though h has at most 2^p-1=7 non-terminal vertices. Moreover, due to the symbolic terminal values, $h_0,...,h_7$, in the diagram[1], there is less opportunity for reduction than an ordinary OBDD with terminal values 0 and 1 only. In this situation, one would prefer trying all possible 2^{2^p}=256 combinations of $h_0,...,h_7$ such that the OBDD of F can be reduce to a reasonable size.

In the following, we will present a novel method to reduce the number of passes from 2^{2^p} exhaustive substitution to 2^p. The method will be illustrated with the case of p=2. In this case the problem becomes to find the boolean function $h=h(v_1,v_2)$ such that

$$h'(f_1,f_2)\,L + h(f_1,f_2)\,H = 0. \quad (3)$$

By Shannon's expansion, we obtain

$$h(f_1,f_2) = f_1'f_2'h_0 + f_1'f_2\,h_1 + f_1\,f_2'h_2 + f_1\,f_2\,h_3,$$
$$h'(f_1,f_2) = f_1'f_2'h_0' + f_1'f_2\,h_1' + f_1\,f_2'h_2' + f_1\,f_2\,h_3', \quad (4)$$

where $h_0=h(0,0)$, $h_1=h(0,1)$, $h_2=h(1,0)$, $h_3=h(1,1)$.
Substituting (4) into (3), then (3) is equivalent to

$$f_1'f_2'L\,h_0' + f_1'f_2'H\,h_0 = 0$$
$$f_1'f_2\,L\,h_1' + f_1'f_2\,H\,h_1 = 0$$
$$f_1\,f_2'L\,h_2' + f_1\,f_2'H\,h_2 = 0$$
$$f_1\,f_2\,L\,h_3' + f_1\,f_2\,H\,h_3 = 0.$$

The value of h_0 can be assigned to be 0 (1) if and only if $f_1'f_2'L=0$ ($f_1'f_2'H=0$). And h_1, h_2, and h_3 can be similarly assigned to be 0 (1) under corresponding conditions. If all h_i's are solvable, then all possible solutions of function h can be found. Otherwise, the gate is not f-correctable.

The above discussions can be easily extended to cases with more than two fan-ins. For a function f, denote $f^1=f$ and $f^0=f'$. Then the result can be summarized in the following.

PROPOSITION 4:

In Eq.2, a solution function $h=h(v_1,v_2,...,v_p)$ of p variables exists if and only if for any given binary values of $t_1,t_2,...,t_p$,

$$f_1^{t1} f_2^{t2} ... f_p^{tp} L = 0 \text{ or } f_1^{t1} f_2^{t2} ... f_p^{tp} H = 0.$$

And for any values $t_1,t_2,...,t_p$, the value of h can be determined as follows:

$h(t_1,...,t_p)$ can be assigned to be 0 iff $f_1^{t1} f_2^{t2} ... f_p^{tp} L = 0$
$h(t_1,...,t_p)$ can be assigned to be 1 iff $f_1^{t1} f_2^{t2} ... f_p^{tp} H = 0$.

Note that conditions

$$f_1^{t1} f_2^{t2} ... f_p^{tp} L = 0$$

and

$$f_1^{t1} f_2^{t2} ... f_p^{tp} H = 0$$

in Proposition 4 are equivalent to

$$f_1^{t1} f_2^{t2} ... f_p^{tp} L_j = 0 \quad \text{for all } j=1,2,..,m$$

and

$$f_1^{t1} f_2^{t2} ... f_p^{tp} H_j = 0 \quad \text{for all } j=1,2,..,m.$$

The latter conditions can be checked for each outputs more efficiently than the original ones since the complexity of boolean operations is strongly dependent to the sizes of their operands[9].

By Proposition 4, our method reduces the searching of h from 2^{2^p} passes to 2^p passes. Moreover, when some h_i does not exist, then the gate is not f-correctable and the process can be terminated immediately without further computation.

The above method of f-correction is in fact an approach of divide-and-conquer. Consider the equality

$$h(f_1,...,f_p) = \sum f_1^{t1} f_2^{t2} ... f_p^{tp}\, h(t_1,t_2,...,t_p),$$

where the summation is over all possible boolean values of $t_1,t_2,...,t_p$. We use each function product, $f_1^{t1} f_2^{t2} ... f_p^{tp}$, with L and H to determine the corresponding $h(t_1,t_2,...,t_p)$; while [1] computes the sum of all products $f_1^{t1} f_2^{t2} ... f_p^{tp}\, h(t_1,t_2,...,t_p)$, and then uses L and H to solve all the unknown values. In terms of binary decision diagrams, the number of vertices in our method is bounded by the maximal size of function products which is drastically smaller than the size of sum of these products. This fact can be demonstrated in the f-correction process for a typical 3-input gate in a 32-bit adder. In our experiment, the maximal size of products is only 45, while the sizes of increasingly partial sums of the 8 products are 46, 135, 316, 674, 1355, 2657, 5301, and 10589 after reduction. The resultant size of the sum, 10589, is 200 times greater than that of any product. The contrast is even more significant for larger gate-fanin number p.

V. CONCLUSION

Diagnosis is a process to locate and correct design errors of an erroneous system. The problem of automatic diagnosis of digital circuits with efficiency had been studied in this paper. Based on the screen-then-correct approach of [1], our automatic diagnosis algorithm consists of two phases: to find possibly correctable gates, and then try to solve the gate function for each of these gates. The solution, if exists, is the desired gate function to correct the circuit errors.

Two improvements over the method of [1] had been developed to enhance the efficiency of diagnosis. Specifically, the dominance relation in circuit topology had been utilized to reduce the search space of possibly correctable gates. In our experiment, the search space is reduced to about 1/2. A novel divide-and-conquer technique to determine the correct gate function had also been proposed. As demonstrated in Section IV, the size of functions to manipulate can be smaller than that of [1] by orders of maginitude for large circuits. The diagnosis algorithm had been implemented in our new version of verification and diagnosis system VVDS[5]. The improved diagnosis method can also be extended to the multiple-error case.

REFERENCES

[1] J. C. Madre, O. Coudert, & J. P. Billon, "Automating the Diagnosis and the rectification of Design Errors with PRIAM," *ICCAD*, pp.30-33, 1989.

[2] G. Odawara, M. Tomita, O. Okuzawa, T. Ohta, & Z. Zhuang, "A logic Verifier Based on Boolean Comparison," *Proc. 23th Design Automatic Conference*, pp.208-214, 1986.

[3] R. Reiter, "A Theory of Diagnosis from First Principles," *Artificial Intelligence*, No. 32, Elsevier Science Publishers, pp.57-95,1987.

[4] S. C. Lee, *Digital Circuit and Logic Design*, Prentice-Hall, 1976

[5] H. T. Liaw, K. T. Tran, & C. S. Lin, "VVDS: A Verification/ Diagnosis System for VHDL," *Proc. 26th Design Automatic Conference*, pp.435-440, 1989.

[6] S. Rudeanu, *Boolean Functions and Equations*, North- Holland Publishing Co. Amsterdam, 1974.

[7] A. V. Aho, J. D. Ullman, *Principles of Compiler Design*, Addison-Wesley, 1977.

[8] D. Harel, "A Linear Time Algorithm for Finding Dominators in Flow Graphs and Related Problems," *Proc. of 17th Annual ACM Symposium on the Theory of Computing*, pp. 185-194, 1985.

[9] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. on Computers*, vol. C-35, No. 8, pp.677-691, August 1986.


```
ALGORITHM : DIAGNOSIS FOR SINGLE ERROR

1.  C := the intersection of fan-in cones of all
    erroneous   primary outputs ;
    Mark all gates in C s-correctable ;
2.  FOR each gate V in C (in backward topological
    order) DO {
3.      IF ( idom(V) is not s-correctable ) THEN {
4.          mark V to be not s-correctable;
5.      }
6.      ELSE {  /* Now test s-correctability */
7.          Replace the output value of V by a new
            variable z, and compute the modified
            circuit functions Fy(x,z) for all y in
            APO, where APO is the set of primary
            outputs in the fan-out cones of V.
8.          FOR each y in APO DO {
9               Ly := Fy(x,0) [xor] Sy(x) ;
10.             Hy := Fy(x,1) [xor] Sy(x) ;
11.         }
12.         IF ( L H != 0 ) THEN {
13.             Mark   the   gate   V   to   be   not
                s-correctable ;
14.         }
15.         ELSE { /* Now test f-correctability */
16.             Try to correct gate V by some
                function h(u) ;
17.             IF (success) THEN {
18.                 RETURN (V,h(u)) ;
19.             }
20.         }
21.     }
22. }
23. RETURN("This circuit can't be corrected by
    single-gate modification.");
```

Fig. 1. Algorithm

TABLE 1. Dominance Ratios

circuits	#PI	#gate	#PO	#tree	dom_ratio
Add32	65	221	32	125	57%
ALU32	70	632	34	255	40%
C499	41	243	32	91	37%
C880	60	443	26	121	27%
C1355	41	587	32	91	16%
C5315	178	2485	123	502	20%

#PI, #PO, #gate: The number of primary inputs, primary output and gates

#tree: the number of dominance trees
dom_ratio = #tree / #gate

TABLE 2. Results

cases	Tv (sec)	Ts (sec)	#s	Ts/#s (sec)	Tf (sec)	#f	Tf/#f (sec)
Add32a	1.22	1.95	1	1.95	0.05	1	0.05
Add32b	1.30	1.58	1	1.58	0.01	1	0.01
Add32c	0.58	1.30	1	1.30	0.18	1	0.18
Add32d	0.83	4.13	4	1.03	0.20	2	0.10
Add32e	1.25	4.13	4	1.03	0.43	3	0.14
ALU32a	140	1824	10	182.4	0.02	1	0.02
ALU32b	140	1464	9	162.7	0.03	1	0.03
ALU32c	122	1456	8	182.0	0.02	2	0.01
ALU32d	135	966	7	138.0	10.23	5	2.05
ALU32e	136	565	3	188.3	14.05	3	4.68
C499a	1406	46252	18	2570	344	3	115
C499b	1072	40043	14	2860	869	7	124
C499c	899	3081	1	3081	36	1	36
C499d	948	231	1	231	686	1	686
C499e	939	65	1	65	69	1	69
C880a	56	5656	28	202	41	3	13.7
C880b	97	1315	5	263	615	5	123
C880c	91	294	14	21	173	11	15.7
C880d	92	2262	11	206	898	9	99.9
C880e	92	0.03	3	0.01	0.02	3	0.01
C1355a	2318	91175	19	4799	924	4	231
C1355b	2186	92130	23	4006	799	7	114
C1355c	5333	46547	13	3581	2830	5	566
C1355d	2000	111	1	111	3586	1	3586
C1355e	1989	505	4	126	215	3	72
C5315a	183	2635	13	203	1.61	4	0.40
C5315b	173	4597	17	270	1.35	5	0.27
C5315c	182	3.2	1	3.2	2.30	1	2.30
C5315d	181	580	4	95	10.3	1	10.3
C5315e	181	58.5	1	58.5	2.6	1	2.6

Tv: Time on verification

Ts(Tf): Total time on testing s(f)-correctability, including the time on computing L_j and H_j.

#s(#f): The number of gates on tesing s(f)-correctability

Ts/#s(Tf/#f): The average time on testing s(f)-correctability for a gate

* All CPU-time data are obtained on SPARC server 390.