A Redundant Fault Identification Algorithm with Exclusive-OR Circuit Reduction

Miyako Tandai
General Purpose Computer Div.
Hitachi, Ltd.
Hadano-shi, Japan, 259-13
Tel: 0463-88-1311
Fax: 0463-87-5904
e-mail: mtandai@kanagawa.hitachi.co.jp

Takao Shinsha
General Purpose Computer Div.
Hitachi, Ltd.
Hadano-shi, Japan, 259-13
Tel: 0463-88-1311
Fax: 0463-87-5904
e-mail: tshinsha@kanagawa.hitachi.co.jp

Abstract–This paper describes a new redundant fault identification algorithm with Exclusive-OR circuit reduction. The experimental results using this algorithm with a FAN-based test pattern generation algorithm show nearly 100% fault coverage for complex arithmetic logic circuits. Moreover we achieved 99.99% fault coverage applying this algorithm with a weighted random pattern generator to the LSIs (100-450 kgates) of Hitachi MP5800 mainframe computer.

1. INTRODUCTION

With the increase in size and complexity of logic circuits in recent years, fault diagnosis is becoming more and more difficult. To overcome this difficulty, scan-design techniques are generally used [1]. Using scan-design techniques, test pattern generation (TG) for sequential circuits can be replaced with that for combinational circuits. This allows us to use TG algorithms for combinational circuits, such as the D-algorithm [2], PODEM [3], and FAN [4] which provide high fault coverage. However, there still remain aborted faults for which test patterns cannot be generated and redundancies cannot be proved. The existence of aborted faults may lower the reliability of an LSI. Especially, for a very large-scale computer which consists of many LSIs, almost 100% fault coverage for each LSI is necessary to obtain satisfiable reliability.

In order to generate test patterns or prove redundancies for such hard faults in combinational circuits, several approaches have been proposed. One is SOCRATES which accelerates the TG and the redundancy identification (RI) processes with improved implication and improved sensitization procedures [5]. Another is Nemesis which proceeds TG and RI using the Boolean satisfiability method [6]. We have also proposed an RI algorithm, REDUCT (REDUndant fault identification algorithm using Circuit reduction Techniques) which is aimed at RI and TG for hard faults [7]. The methodology of REDUCT is a new one which reduces the complexity of circuits making TG and RI easier. We have applied N^2-V (the N^2-Valued test pattern generation algorithm) [8] which is a FAN-based TG algorithm and then applied REDUCT which handles all of the aborted faults of N^2-V, to the ISCAS85 benchmarks [9], and have obtained 100% fault coverage for all of the benchmarks [7].

The TG problem is NP-hard and the RI problem is co-NP-complete [10]. The difficulties of the problems are caused by a large number of reconvergences, head lines [4], etc. in combinational circuits. To determine the effects of reconvergences and head lines in a combinational circuit, REDUCT handles the redundant fault identification problem by transforming the given circuit into another. Given combinational circuit X and the problem "Is fault f redundant in X or not?", REDUCT transforms them into circuit Z consisting of NOR-gates and primary inputs (PIs) and an equivalent problem "Is the output of Z equal to 0 for any input pattern or not?". REDUCT next reduces circuit complexity of Z, using four circuit reduction techniques. And finally, REDUCT tries to justify the logic value 1 of the reduced circuit output by backtracing and backtracking, and determines whether or not an input pattern exists that causes the circuit output to be 1.

We have also applied N^2-V and REDUCT to several arithmetic logic circuits, but the fault coverage was not as good as with other types of circuit. It is difficult in general to achieve high fault coverage for an arithmetic logic circuit. An arithmetic logic circuit includes many EOR-logics, so that the complexity of the circuit increases. REDUCT which constructs the circuit with only NOR-gates and primary inputs (PIs), cannot identify EOR-logics in the circuit nor the complexity caused by EOR-logics. Fig. 1 shows an example of this case. In Fig. 1, A, B, C, G, H, and K are NOR-gates, and the output values of G and H are 0. Because G = EOR(K, C) = EOR(EOR(A, B), C) = EOR(A, B, C) and H = EOR(A, C), B = EOR(G, H) = 0. But not being able to identify these EOR relations, REDUCT cannot infer that the output value of B is 0.

To overcome this, it is necessary to identify EOR-logics and to see how these EOR-logics relate to one another. Therefore, in this paper we propose an improved REDUCT which identifies EOR-logics, reduces the number of neighbours of EOR-logics, and does improved implication using the properties of EOR-logics during the justification procedure.

We applied N^2-V and the original or improved REDUCT to some arithmetic logic circuits. Using N^2-V and the improved REDUCT, we obtained higher fault coverage than was obtained using N^2-V and the original REDUCT. We also applied the improved REDUCT after applying a weighted random pattern generator (WRPG) to the LSIs of Hitachi MP5800 mainframe computer, and achieved 99.99% fault coverage.

This paper begins with a description of the algorithm and the circuit reduction techniques used to reduce circuit complexity, of the original REDUCT in Sec. 2. We then describe in Sec. 3 the improvement of REDUCT (EOR-logic reduction and improved implication). Experimental results are given in Sec. 4 and Sec. 5 concludes the paper.

2. ORIGINAL REDUCT

2.1. ALGORITHM

In this section, we will describe the original REDUCT algorithm.

The flowchart in Fig. 2 outlines the REDUCT algorithm. First, for each pair (*f*, *C*), where *f* is an undetected fault in combinational logic *X* and *C* is a cone containing *f*, related regions are determined. Here, a cone is a single-output sub-circuit of *X*, whose primary output (PO) is one of the POs of *X*, and the related region for (*f*, *C*) is the limited region such that the redundancy of *f* in this region is equivalent to that in *C*. Fig. 3 shows an example of related regions. The circuit in Fig. 3 has three cones X_1, X_2, and X_3. Assume that either a s-a-0 or a s-a-1 fault on line l_1 (l_2) has been detected at either O_1 or O_2(O_3). Fig. 3(a) shows the related region $R(f, X_1)$ (= $R(f, X_2)$) for fault *f* and X_1 (X_2). Fig. 3(b) shows the related region $R(f, X_3)$ for *f* and X_3. After related region determination, it is enough to treat $R(f, X_1)$ and $R(f, X_3)$ instead of X_1, X_2, and X_3 for *f*.

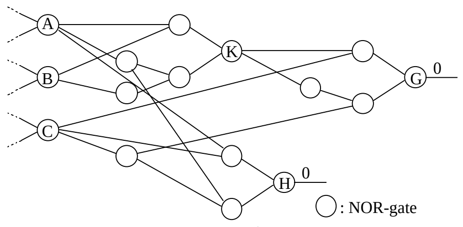
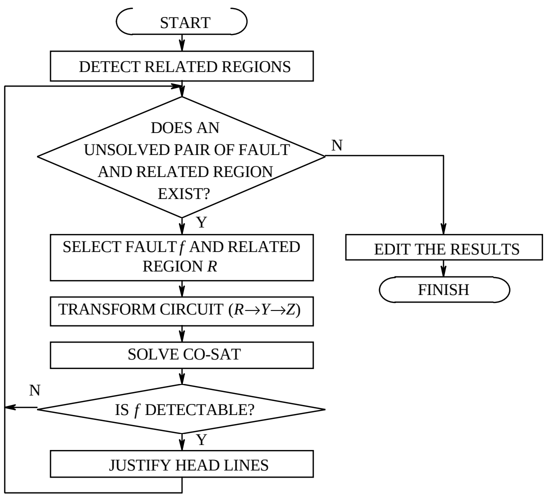


Fig. 1. Example of EOR-logics



START
DETECT RELATED REGIONS
DOES AN UNSOLVED PAIR OF FAULT AND RELATED REGION EXIST?
N
Y
SELECT FAULT *f* AND RELATED REGION *R*
EDIT THE RESULTS
FINISH
TRANSFORM CIRCUIT ($R \to Y \to Z$)
SOLVE CO-SAT
N
IS *f* DETECTABLE?
Y
JUSTIFY HEAD LINES

Fig. 2. REDUCT algorithm

Then, for each pair of faults and cones, the related region *R* is transformed into a single-output circuit *Y* for the co-SAT (co-SATisfiability problem) as described in the flowchart in Fig. 4, and circuit *Y* is transformed into an equivalent circuit *Z* consisting of NOR-gates and PIs. Fig. 4(a) shows circuit *R*, with a fault *f* on line l_f. If *f* is s-a-1, let R' be a sub-circuit of *R* whose PO is l_f. If *f* is s-a-0, let R' be a circuit obtained by adding a NOT-gate to PO l_f of the above R'. Fig. 4(b) shows R'. Let R'' be a sub-circuit of *R* obtained by deleting from *R* the gates and the lines, which cannot be accessible to the circuit output without passing through l_f. Fig. 4(c) shows R''. Next, let $R''(0)$ and $R''(1)$ be the circuits assigned 0 and 1 to l_f of R'', respectively. Circuit *Y* is then constructed with R', $R''(0)$, $R''(1)$, an ENOR-gate *E*, and an output NOR-gate *O* as shown in Fig. 4(d).

Then the RI problem "Is *f* redundant in *R*?" for fault *f* and the related region *R* is equivalent to the co-SAT "Does the output of *Y* (*Z*) become 0 for any input pattern?" for circuit *Y* (*Z*). The solution of the co-SAT may be one of the following: yes (i.e., *f* is redundant in *R*), no (i.e., *f* is detectable in *R*), or unknown (i.e., *f* cannot be identified as either redundant or not because the computational effort to solve the problem exceeds the limit). If the solution is no, then a test pattern for *f* is generated by justifying all the logic values assigned to head lines while solving the co-SAT.

Fig. 5 outlines the algorithm for solving the co-SAT process in Fig. 2. Circuit *Z* is reduced by using the circuit reduction techniques. The original REDUCT has four circuit reduction techniques (binding identical sub-logics (BI), renewal of head lines (RH), reduction 1 (R1), and reduction 2 (R2)). If the co-SAT has not been solved during circuit reduction, then justify the logic value 1 of the reduced circuit output by backtracing and backtracking. If the justification fails for all possible backtracking, the solution of the co-SAT is yes.

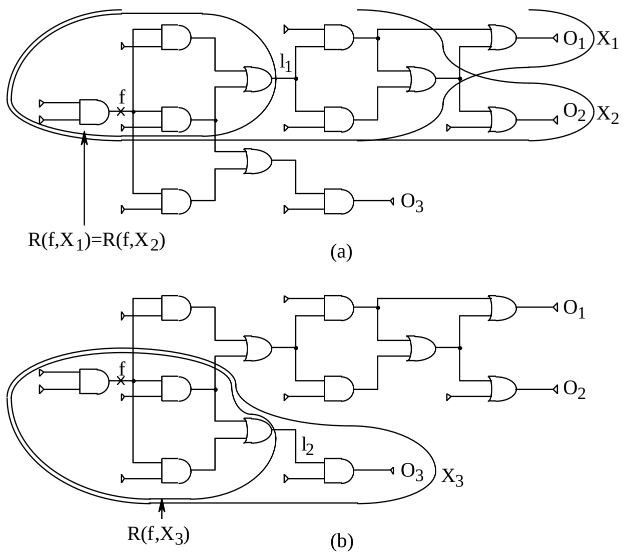


Fig. 3. Related regions
(a) Related region for (X_1, *f*) and (X_2, *f*)
(b) Related region for (X_3, *f*)

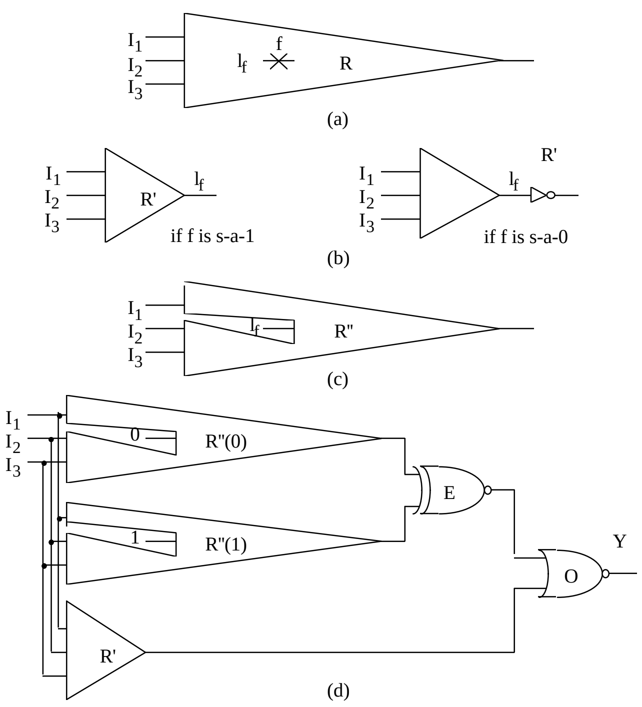


Fig. 4. Transformation of *R* into *Y*
(a) Circuit *R* (b) Circuit *R'* (c) Circuit *R''* (d) Circuit *Y*

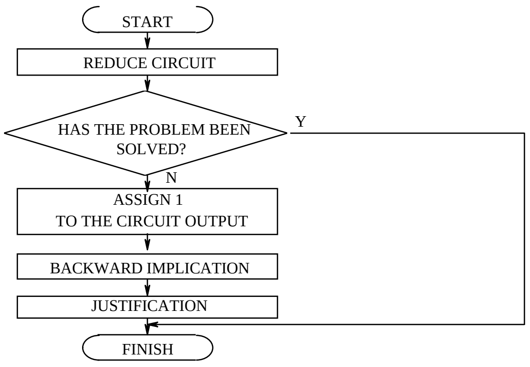


Fig. 5. Algorithm for solving the co-SAT

2.2. CIRCUIT REDUCTION TECHNIQUES

In this section the four reduction techniques to reduce the complexity of the circuit is described (circuit reduction processing in Fig. 5). BI and R1 reduce reconvergences, and RH and R2 decrease the number of head lines, and ER reduces the number of neighbours of EOR-logics. Before describing these techniques, we introduce independency and dependency of gates in a circuit. The dependency of gates is important for SAT (SATisfiability problem), because higher dependency generally makes it more difficult to solve SAT.

Let G_1, G_2, ..., and G_n be gates or PIs in a circuit, whose output lines are l_1, l_2, ..., and l_n, respectively. Here, G_1, G_2, ..., and G_n are defined to be independent (or l_1, l_2, ..., and l_n to be independent) if for any n-tuple vector $v=(v_1, v_2, ..., v_n)\in\{0, 1\}^n$ there exists an input pattern which controls the logic value of l_i to be v_i for all i. If G_1, G_2, ..., and G_n are not independent, we say that G_1, G_2, ..., and G_n are dependent (or l_1, l_2, ..., and l_n are dependent). All the PI lines and all the head lines provide examples of independent lines. Of the four reduction techniques, BI handles special dependent cases, RH tries to find a set of independent gates located nearer to the circuit output than the head lines, and R1 finds dependent gates.

A. *Binding identical sub-logics (BI)*

A typical example of dependent gates is a pair of gates that have identical sub-logic. In this case, the two gates can be bound. BI is essentially the same as the alignment of [11].

Fig. 6 shows an example of BI. In Fig. 6(a), NOR-gates *A* and *B* express the same sub-logic, because *A* and *B* have the same source gates *C* and *D*. Fig. 6(b) shows the circuit after gates *A* and *B* have been bound into one gate *A*.

B. *Renewal of head lines (RH)*

RH is a technique to reduce head lines. First we define an independent cutset of circuit *Z* as a set *G* of lines satisfying the following conditions.
(1) Lines in *G* are independent.
(2) Any path from any PI to the output of *Z* passes through at least one line in *G*.
It is clear that the set of all the PI lines is an independent cutset and so is the set of all head lines. The concept of an independent cutset resembles that of the basis node of TOPS [12]. Unlike the basis node, however, the independent cutset takes into account the precise controllability of lines. If there exists an independent cutset whose lines are head lines or are located nearer to the output of *Z* than the head lines, the lines in this independent cutset can be treated as head lines. Using this property RH reduces head lines.

Fig. 7 shows an example of RH. In Fig. 7(a), the output lines of PIs I_1, I_2, I_3, and I_4 are head lines and the output line of I_1 has fan-out branches. Fig. 7(b) shows the circuit after implication of the assignment of 0 to the output of I_1. In Fig. 7(b), the outputs of NOR-gate *A*, *B*, and *C* can be controlled by the values of the output lines of I_2, I_3, and I_4, which are head lines. Fig. 7(c) shows the circuit after cutting lines whose logic values are 0 or 1 and then deleting gates and PIs unconnected to a sink gate. The output line of *D* is the head line renewed from the output lines of PIs I_1, I_2, I_3, and I_4.

C. *Reduction 1 (R1)*

For a sub-logic expressed by a gate in a circuit, a case exists where the sub-logic can be expressed more simply. A

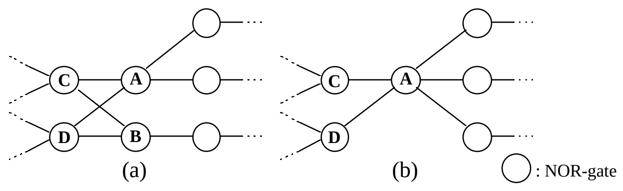


Fig. 6. Binding identical sub-logics
(a) Initial situation (b) Binding *A* and *B*

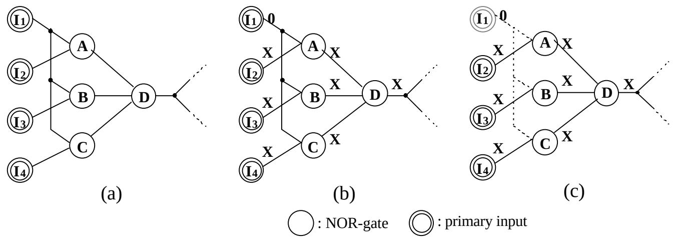


Fig. 7. Renewal of head lines
(a) Initial situation
(b) Implication of the assignment of 0 to the output of *I*
(c) Final situation: line cut and gate deletion

technique called R1 that simplifies gate expressions in the circuit is described in this section. R1 first investigates the relation of the objective gate *R* and the gates in the fan-in side of *R*, then using this relation reduces the fan-in side logics of *R*. We describe R1 using the two examples in Fig. 8 and Fig. 9.

Fig. 8 shows an example of R1. In Fig. 8(a), NOR-gate *R* is an objective gate. Fig. 8(b) shows the circuit after implication of the assignment of 1 to the output of *R*. The result of implication shows that the output value of *R* is 1 only if the output values of *B*, *C*, and *I* are all 0. Fig. 8(c) shows the circuit after replacing the sub-circuit surrounded by *R*, *B*, *C*, and *I* by a simple logic NOR(*B*, *C*, *I*).

Fig. 9 shows another example of R1. In Fig. 9(a), NOR-gate *R* is an objective gate. Fig. 9(b) shows the circuit after implication of the assignment of 1 to the output of *R*. During implication, a contradiction occurs. Thus, the output value of *R* must be 0. Fig. 9(c) shows the circuit assigned 0 to the output of *R*.

D. *Reduction 2 (R2)*

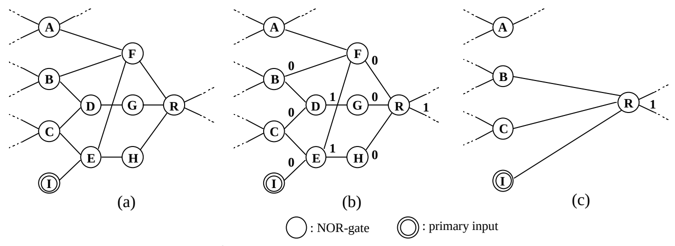


Fig. 8. First example of reduction 1
(a) Initial situation
(b) Implication of the assignment of 1 to the output of *R*
(c) Final situation: deletion of sub-circuit surrounded by *B*, *C*, *I*, and *R*, and connection of *R* with *B*, *C*, and *I*

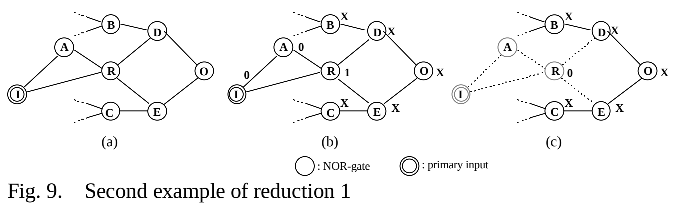


Fig. 9. Second example of reduction 1
(a) Initial situation
(b) Implication of the assignment of 1 to the output of *R*
(c) Final situation: Implication of the assignment of 0 to output of *R*, line cut, and gate deletion

R2 reduces head lines. Let *C* be an one-output NOR constructed circuit and *h* be a head line in *C*. In general, there are several paths from *h* to the circuit output and the effect of *h* on the circuit output is very complex. However, in special cases, *h* has a useful property which is stated in the theorem below.

Theorem 1. Assume that each path from *h* to the output of circuit *C* has an odd number of NOR-gates. Then circuit *C′* obtained by the assignment *h*=0 has no input pattern which causes the output of *C′* to be 1, if and only if *C* has no such input pattern.

On the other hand, assume that each path from *h* to the output of *C* has an even number of NOR-gates. Then circuit *C′* obtained by the assignment *h*=1 has no input pattern which causes the output of *C′* to be 1, if and only if *C* has no such input pattern.

Proof (sketch in the case of odd number of NOR-gates): It is obvious that if *C* has no input pattern which causes the output of *C* to be 1, then *C*’ has no such input pattern. Assume if *C*’ has no input pattern which causes the output of *C*’ to be 1 but *C* has one such input pattern *T*. The logic value of *h* must be 1 under *T*. When the logic value of *h* is changed from 1 to 0, the output value of *C* becomes 0. This logic value propagation means that under *T*, there is a path from *h* to the output of *C* on which there emerges 0 and 1 logic values alternatively, which is impossible because there is no even number of NOR-gates path from *h* to the output of *C*. Q.E.D.

R2 reduces head lines using the properties stated in Theorem 1.

Fig. 10 shows an example of R2. In Fig. 10(a), from the head line *h* to the output of *O* (circuit output gate) there are two paths, *A-C-O* and *B-D-O*, each of which consists of three NOR-gates. Fig. 10(b) shows the circuit assigned 0 to *h*.

3. IMPROVEMENT OF REDUCT

The improved REDUCT has EOR-logic reduction technique (ER) added to the original four techniques BI, RH, R1, R2 for circuit reduction, and does improved implication for EOR logics in the circuit. ER reduces the number of neighbours of EOR-logics

3.1. EOR-LOGIC REDUCTION (ER)

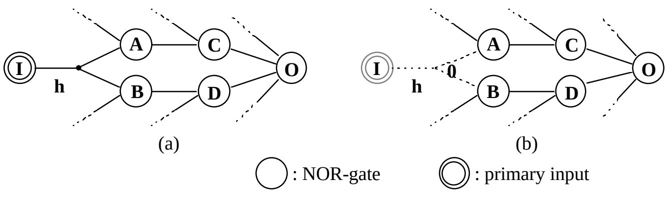


Fig. 10. Reduction 2
(a) Initial situation
(b) Implication of the assignment *h*=1, line cut, and gate deletion

ER identifies product-sum sub-circuits expressing EOR(ENOR)-logics in the circuit and reduces the neighbours of the EOR(ENOR)-logics, to reduce the complexity caused by EOR(ENOR)-logics.

Fig. 11 shows an example of identifying EOR-logic. In Fig. 11(a), G is the objective NOR-gate. The output logic of G is equal to the product-sum logic expressing EOR(I_1, I_2, I_3). Fig. 11(b) shows the circuit after replacing the product-sum logic with an EOR-gate G.

There are two ways to reduce the number of neighbours of an EOR-gate. One is to reduce the fan-in inverters of an EOR(ENOR)-gate and the other is to reduce the fan-in EOR(ENOR)-gates of an EOR(ENOR)-gate. Fig. 12 and Fig. 13, respectively, show examples of the former and latter reductions.

In Fig. 12, G is an EOR-gate and I_1, I_2, I_3, A, B, and C are NOR-gates. In Fig. 12(a1), G = EOR(I_1, I_2, C) = EOR(A, B, C), hence I_1 and I_2 can be reduced as in Fig. 12(b1). In Fig. 12(a2), G = EOR(I_1, I_2, I_3) = ENOR(A, B, C), hence I_1, I_2, and I_3 can be reduced by changing G from an EOR-gate to an ENOR-gate as in Fig. 12(b2).

In Fig. 13, G and H are EOR-gates and A, B, and C are NOR-gates. In Fig. 13(a1), G = EOR(H, C) = EOR(A, B, C), hence H can be reduced as in Fig. 13(b1). In Fig. 13(a2), G = EOR(H, C) = ENOR(A, B, C), hence H can be reduced by changing G from an EOR-gate to an ENOR-gate as in Fig. 13(b2).

3.2. Improved implication

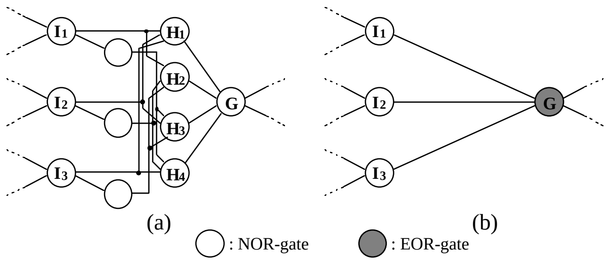


Fig. 11. Example of identifying EOR-logic
(a) Initial situation (b) Final situation

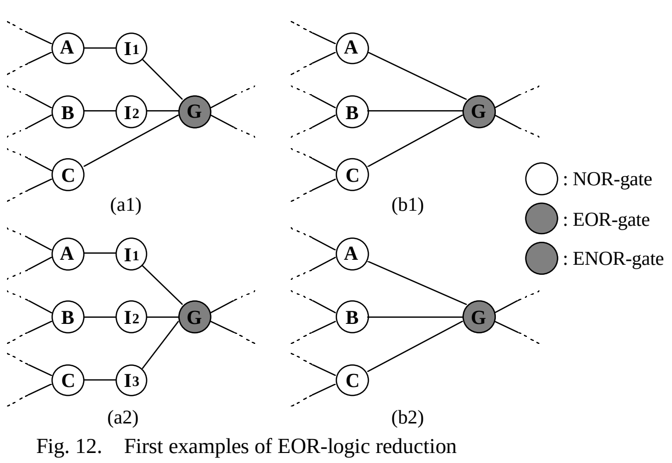


Fig. 12. First examples of EOR-logic reduction
(a1), (a2) Initial situations (b1), (b2) Final situations

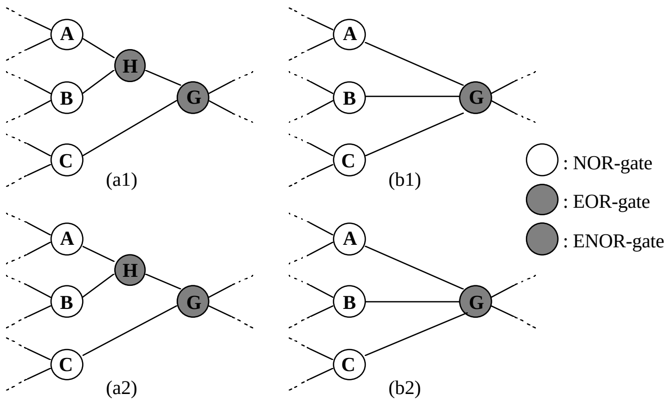


Fig. 13. Second examples of EOR-logic reduction
(a1), (a2) Initial situations (b1), (b2) Final situations

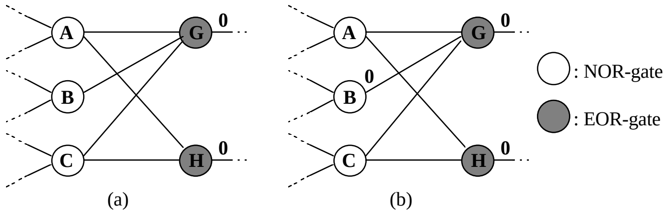


Fig. 14. Example of improved implication
(a) Initial situation (b) Final situation

During the implication following backtracing and backtracking, the improved REDUCT does improved implication using the properties of the EOR-gate. Fig. 14 shows an example of this improved implication. In Fig. 14(a), the output values of EOR-gates G and H are both 0; and A, B, and C are NOR-gates. Because G = EOR(A,B,C) and H = EOR(A,C), we have B = EOR(G,H) = 0 (Fig. 14(b)). If G and H are constructed with only NOR-gates we cannot obtain the information that the output value of B is 0.

4. Experimental results

We applied N^2-V and REDUCT to arithmetic logic circuits containing many EOR-logics. N^2-V is a FAN-based TG algorithm which handles normal values and faulty values separately in order to express the value assignment situation in more detail. We first applied N^2-V to the circuits and then applied REDUCT to the aborted faults for which N^2-V failed to generate test patterns and to prove redundancies. We used a Hitachi M680 mainframe computer for this work. In the results described below, fault coverage is defined as (no. of detected faults $*$ 100) / (no. of assumed faults $-$ no. of redundant faults).

Table 1 shows the results for real combinational arithmetic logic circuits. Using N^2-V and the original REDUCT, we did not obtain 100% fault coverage for any of the nine circuits. But, using N^2-V and the improved REDUCT, we produced higher fault coverage for all of the circuits and for seven of the nine we obtained 100% fault coverage.

Table 2 shows the results for full-scanned sequential complex arithmetic logic circuits of actual VLSIs. The TG system that we used divided the circuit into combinational sub-circuits, and N^2-V and REDUCT were applied to each of them. Using N^2-V and the improved REDUCT, we obtained higher fault coverage than was obtained using N^2-V and the original REDUCT, and the number of aborted faults was much lower.

From the results shown in Tables 1 and 2, we can conclude that the improved REDUCT is very effective in achieving higher fault coverage for arithmetic logic circuits.

We also applied a WRPG with VCLAS (Vector Command array Library Active Simulation: a vectorized LCC (Levelized Compiled Code) fault simulation [13]) and the improved REDUCT to more than 30 LSIs (100-450 kgates) of MP5800. Applying a WRPG with VCLAS only we achieved 99.94% fault coverage on the average, but applying the improved REDUCT after this we achieved 99.99% fault coverage on the average. This result shows that the improved REDUCT is also effective for TG and RI of very large circuits.

5. Conclusion

We refined an RI algorithm REDUCT which was proposed in [7]. This improved algorithm deals with the RI problem by transforming a given circuit into another. Then it uses five circuit reduction techniques to reduce the complexity of the transformed circuit caused by a large number of reconvergences, head lines, and EOR-logics in the circuit. It also proves redundancies and generates test patterns for hard faults of circuits containing many EOR-logics more efficiently than conventional TG algorithms. Using a combination of N^2-V and the improved REDUCT, we obtained higher fault coverage for arithmetic logic circuits than was obtained using N^2-V and the original REDUCT. Also using the improved REDUCT with a WRPG to the LSIs of MP5800 we achieved 99.99% fault coverage.

Acknowledgments

We would like to thank Masayuki Miyoshi for giving us the opportunity to improve REDUCT. Also, we appreciate the experimentation support provided by Jun Matsushima.

TABLE 1. Results by N^2-V and REDUCT (1)

Circuit	Gates	N^2-V+original REDUCT			N^2-V+improved REDUCT		
		Fault Coverage	Aborted Faults	CPU-time (sec.)	Fault Coverage	Aborted Faults	CPU-time (sec.)
a	5243	98.99	1	4731	100.00	0	4633
b	720	99.50	2	155	100.00	0	71
c	5508	99.94	9	15551	99.99	2	14478
d	2860	99.89	2	725	100.00	0	337
e	2917	99.77	2	330	100.00	0	187
f	7419	99.98	1	549	100.00	0	343
g	850	99.60	2	172	100.00	0	109
h	6287	99.98	3	2735	99.99	1	2584
i	1686	99.80	3	478	100.00	0	236

TABLE 2. Results by N^2-V and REDUCT (2)

Circuit	Gates	N^2-V+original REDUCT			N^2-V+improved REDUCT		
		Fault Coverage	Aborted Faults	CPU-time (sec.)	Fault Coverage	Aborted Faults	CPU-time (sec.)
A	22858	99.68	313	2969	99.97	26	2226
B	23650	99.09	810	7977	99.79	183	8366

References

[1] H. Fujiwara, "Logic testing and design for testability", MIT PRESS, p. 284, March 1985.

[2] J.P. Roth, W.G. Bouricius, and P.R. Schneider, "Programmed algorithm to compute tests to detect and distinguish between failures in logic circuits", IEEE Trans. Compt., EC-16, No. 10, pp. 567-580, Oct, 1967.

[3] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits", IEEE Trans. Compt., C-30, No. 3, pp. 215-222, March 1981.

[4] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithm", IEEE Trans. Compt., C-32, No. 12, pp. 1137-1144, Dec. 1983.

[5] M.H. Schulz and E. Auth, "Improved deterministic test pattern generation with applications to redundancy identification", IEEE Trans. Compt., CAD-8, No. 7, pp. 811-816, July 1989.

[6] T. Larrabee, "Test pattern generation using Boolean satisfiability", IEEE Trans. Compt., CAD-11, No. 1, pp. 4-15, Jan. 1992.

[7] M. Tandai, T. Shinsha, T. Nishida, and K. Moriwaki, "REDUCT: a redundant fault identification algorithm using circuit reduction techniques", IEICE Trans. Info. and Systems. E76-D, No. 3, pp. 776-790, July 1993.

[8] M. Tandai, T. Shinsha, and K. Moriwaki, "An N^2-valued test pattern generation algorithm", Trans. Info. Proc. Soc. of Japan, Vol. 30, No. 9, pp. 1211-1218, Sept. 1989 (in Japanese).

[9] F. Brglez and H. Fujiwara, "A neural netlist of 10 combinational benchmark circuits and a target translator in FORTRAN", Proc. IEEE Int. Symp. Circuits and Systems, pp. 679-682, June 1985.

[10] O.H. Ibarra and S.K. Sahni, "Polynomially complete fault detection problems", IEEE Trans. Compt. C-24, No. 3, pp. 242-249, March 1975.

[11] R. Jacobi, P. Moceyunas, H. Cho, and G. Hachtel, "New ATPG techniques for logic optimization", Proc. International Conference on Computer Aided Design, pp. 548-551, July 1989.

[12] T. Kirkland and M.R. Mercer, "A topological search algorithm for ATPG", Proc. of the 24th Design Automation Conf., pp. 502-508, June 1987.

[13] M. Chiang and R. Ralkovic, "LCC simulators speed development of synchronous hardware", Computer Design, pp. 87-90, March 1986.