Redundant Faults

Srinivas Garimella
Dept. of Electrical and Computer Engineering
Auburn University
Auburn, AL 36849-5201
garimsm@auburn.edu
Reviewer: Jonathan M. Harris

Abstract: *Redundant faults* in digital circuits not only result in extra hardware but also mask other faults and may also make undetectable fault detectable. Redundancies increase the chip area, the power consumption, and often propagation delays in the circuit and might even reduce the yield [5]. This paper is a result of literary search for detection and removal of such faults. This paper addresses procedures described in literature to detect and remove redundant faults.

1. Introduction

Faults in digital circuits for which no test can be found are classified as *untestable faults*. *Redundant faults* are those faults that do not change the input-output function of the circuit. For combinational circuits, redundant faults and untestable faults are same, whereas for sequential circuits, redundant faults form a subset of untestable faults [1].

Undetectable faults are the result of unnecessary gates and connections in the circuits. Their existence increases the complexity of the testing process. Undetectable faults do not effect the operation of the circuit as long as another fault does not occur. However, the presence of an undetectable fault may invalidate a test for some other single fault. The occurrence of an undetectable fault may mask an otherwise detectable fault, or make an undetectable fault detectable [2].

Friedman [6] was the first to point out the significance of redundancy during test generation process. Test generation can detect single redundant faults, but can be time consuming. Few procedures for directly identifying and removing undetectable faults in combinational circuits, without test generation, have been proposed in literature. These direct redundant identification techniques can be either *static* or *dynamic*. Static techniques identify the redundancies before test generation, where as dynamic techniques identify them during test generation. A static redundancy identification technique based on circuit structure is presented in [2]. Another static technique based on controllability and observability analysis is described in [7]. The Procedure for removing redundancies after identifying them statically is proposed in [4]. A dynamic redundant identification and removal technique for combinational circuit is described in [8]. A method called Redundant Identification and Removal [*RIDAR*], which makes use of functional specification of the circuit, is described in [9]. This method is capable of removing more than one redundancy at a time and, hence, can reduce CPU time significantly.

The paper is organized as follows: In Section 2, a static Redundancy Identification (RID) technique for combinational circuits as described in [2] is explained in detail. The procedure to remove untestable faults and simplify the circuit as described in [4] is presented in the section 3. The complexity of identifying redundant faults in sequential circuits and work done in this area is described in the next section. The paper is concluded in the last section.

2. Identification of Untestable Faults

Reconvergent fanout is a necessary condition for a redundant stuck-at fault to exist in a circuit. If two or more paths from a line *j* converge at different inputs of a gate *G*, the gate *G* is called reconvergent gate and the line *j* is called the stem of the gate *G*. A structure with a stem and reconvergent gate is called reconvergent fanout.

Some of the terms used in describing the procedure to analyze the reconvergent regions and to identify untestable faults are defined with the help of Figure 1 as follows:

The set of inputs of a reconvergent gate *G* reachable from its stem *s* is defined as *reconvergent input set* of *G* and is denoted as *I (G, s)*. In the circuit of Figure 1, gate *4* is the reconvergent gate and *f* is its stem. I *(f, 4) = h,i.*

A lead *j* is said to be bound by a gate *A* if all paths from j reach primary output(s) through *A*. In the circuit of Figure 1, line *e* is bound by h as *e* has a path to primary output only through gate *2*.

If the stem value v_s = a (1 or 0) implies a controlling value at any of the inputs of the reconvergent gate, then the path formed by backtracing form that input to stem *s* is called a *controlling value path* or *cv-path*. All other paths from stem *s* to its reconvergent gate are called *non controlling value paths* or *n-cv paths*. In Figure 1, since

stem $f = 0$ implies controlling value 0 for the gate *2* and gate *4*, *f-f1-h* is a cv-path. Similarly *f-f2-i* is also a cv-path obtained by assigning $f=1$. There are no n-cv paths for this circuit.

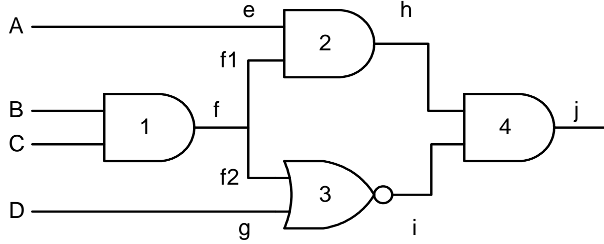

Figure 1. Circuit for detecting untestable faults.

Let v_j be the value on a lead *j*. If v_j = a is necessary condition for producing a value v_k = b, where (a,b) are 0 or 1, then v_j = a is called the essential value for v_k = b. In Figure 1, $f = 0$ implies $h = 0$, therefore, $f = 1$ is the essential value for $h = 1$.

If setting v_j = a produces non-controlling values on all inputs of gate *A* reachable from j, then v_j = a is called the essential sensitizing value for gate *A*. In Figure 1, $f=1$ produces non-controlling value 1 on input *f1* for gate *2,* therefore, $f=1$ is essential sensitizing value for gate 2. Similarly, $f=0$ is essential sensitizing value for gate *3*.

Two sets of procedures are described in [2]: single stem analysis and multiple stem analysis.

2.1 Single Stem Analysis

This procedure is described with examples as follows:

The first step in detecting untestable faults is to find *cv-paths* and *non cv-paths* from a reconvergent gate *G* to a stem *s*. The procedure for determining cv-path is:

Let *G* be the reconvergent gate of stem *s*. For all paths from *s* to *G*, set $s = 0$ if the path is an even parity path or else set $s = 1$. If forward implication produces controlling values on any of inputs of gate *G*, then trace back from all these inputs to stem *s*, to identify all the cv- paths. All the remaining paths are non cv-paths.

If there are no cv-paths, this implies that there are no undetectable faults in the circuit. If there are cv-paths then we continue the analysis.

For every stems *s*, which produces cv-paths, we determine N_f, number of fanout branches that lie on cv-paths. We assign value $v_s = 0$ and determine all the lines which get assigned a value v_j = j by forward implication. Next, we define $v_s = 1$ as essential value for v_j = ~j (inverse of *j*). We repeat the above process by assigning $v_s = 1$. For the circuit in Figure 1 N_f=2. We find that $f=1$ is essential value for *f1, h,* $j = 1$ and $f=0$ is essential value for $f2=0$ and *i,* $j=1$. Also we can find that $f=0$ is essential sensitizing value for gate 3 and $f=1$ is essential sensitizing value for gate 2.

Undetectable faults are found by the following rules:

1) For every gate *A* for which v_s is an essential sensitizing value, both stuck-at faults on every lead *i* bound by *A* but not reachable form *s* are undetectable. For the circuit in Figure 1, $f=1$ is essential sensitizing value for gate *2*. According to the above rule, both stuck-at faults on lead *e* are undetectable. This is so because $f=1$ would allow for the fault on lead *e* to be activated, but the fault cannot be propagated because of the cv-path *f-f2-i*. Similarly both stuck-at faults on lead *g* cannot be detected.

2) If N_f=1, stuck-at- ~v_j on any line *j* for which v_s is essential for setting *j* to v_j is undetectable, if it is not on a cv-path and is bound by reconvergent gate *G*. If $N_f > 1$, stuck-at- ~v_j on any line *j,* bound by *G* and for which v_s is essential for setting j to v_j , is undetectable. For the circuit in Figure 1, N_f=2, leads *f1, f2, h, i* are bound by reconvergent gate *4*. Since $v_s = 1$ is essential for $f1=1$and $h=1$, *f1* s-a-0 and *h* s-a-0 are undetectable. If NOR gate 3 in Figure 1 is replaced by an OR gate, then $N_f = 1$ and the essential value for $i=0$ would be $f=0$. According to rule 2, *i* s-a-1 is undetectable since it is not on a cv-path. This is so because setting $f2=0$ for activating the fault would not allow the propagation of fault as cv-path *f1-h* would be activated.

3) If v_s =0 and v_s =1 would produce same value at the output of reconvergent gate G, then if *s* is bound by *G,* both stuck-at faults on the stem s are undetectable. Faults on any lead *j*, such that all paths from *j* to any primary output that pass through *s* are also undetectable. This is so because the same output for the reconvergent gate *G* implies existence of at least two cv paths such that one is of even parity and the other is of odd parity. Therefore propagation of the fault through one cv path would be affected by the other cv-path. In Figure 1, $j=1$ for $f=0, 1$. Therefore stuck-at-faults on lead *f,* and hence on *f1, f2, B, C* are undetectable. If v_G is the output of *G* for controlling value *c* on any of its input, then output of *G* s-a- v_G is undetectable. If v_G is an essential sensitizing value for any gate A, then both stuck-at faults on any lead *i* bound by A and not reachable from G are undetectable. For the circuit in Figure 1, since the value of *h* = 0 is independent of *f*, *G* s-a-0 is undetectable.

After every fault that is undetectable is identified, all equivalent faults are identified and marked as undetectable.

If a reconvergent gate has multiple stems, multiple stem analysis as described in [2] is carried out. This analysis is similar to single-stem analysis, but combination of stems is analyzed at each step. Multiple-stem

analysis is not discussed in this paper. The analysis was done on some of the ISCAS Benchmark circuits and results are as shown in Table 1. We see that most of these circuits have reconvergent gates with multiple stems and hence multiple-stem analysis detects more faults.

Circuits	c1355	c1908	c2670	c3540	c5315
Single stem	0	6	31	112	22
Multiple-stem	8	9	106	137	59

Table 1. Redundant faults detected in selected benchmark circuits

3. Removal of Untestable Faults

Once all undetectable faults are identified, the circuit has to be simplified by eliminating such faults. When some redundant faults are removed, other redundant faults may become irredundant and some irredundant faults may become redundant. If a redundant fault is removed as soon as it is identified, it is not possible to make a previously identified redundant fault irredundant and hence we have to find only newly introduced redundancies. A procedure to simplify the circuit by eliminating redundant faults by taking into account newly introduced redundancies as described in [4] is discussed below.

If a fault *j* stuck-at-a is undetectable, then disconnecting the lead *j* from it's driving gate and setting it to 'a' will not modify the input-output function of the circuit. If *j* stuck-at-a is undetectable and v_j =a implies $v_{ki} = b_i$ for a set of lines $\{k_i\}$, then removing the farthest line k_i on each path from *j* and setting it to b_i will not effect the circuit. If this procedure produces constant values at the primary inputs, then forward implication can be done simultaneously on the primary inputs to simplify the circuit.

Forward implication would stop when a non-controlling value is produced at an input to any gate. Setting a non-controlling value at an input of the gate is equivalent to disconnecting that input lead from that gate. If that leads to single-input gates, then those gates can be replaced with an inverter or a direct connection depending on whether that gate inverts the input or not. All the gates with no paths to the output are then removed from the circuit.

If constant values are produced at the outputs of reconvergent gates, then forward implication can be carried out and redundant gates and connections can be removed following the same procedure as for constant primary inputs. Circuit modification can then affect detectability of other faults. So the circuit has to be reanalyzed. If an input to gate *A* is removed, then any stem from which *A* can be reached must be reanalyzed to detect any undetectable faults. We analyze the circuit for simplification after levelizing the circuit. We start analyzing in reverse level order. The procedure is explained with the help of Figure 2 as follows:

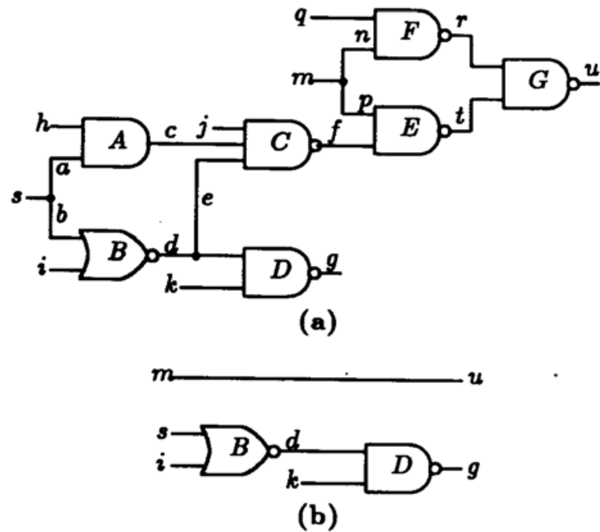

Figure 2. Circuit for Redundancy Removal

Circuit	Total Red. Faults	Lines Removed	Gates Removed	Redundant faults removed	Red. Faults Left
C1355	8	0	0	0	8
C1908	9	11	3	5	3
C2670	117	63	28	34	85
C3540	137	53	9	36	105
C5315	59	38	12	26	35

Table 2. Faults removed in selected benchmark circuits

After assigning levels to the gates in circuit in Figure 2(a), we start analyzing the stem *m* and its reconvergent gate *G* as it has the highest level. We see that there are no cv-paths and hence no undetectable faults. Then we move to analyze stem *s*. Since value of *f* is independent of *s, f* can be disconnected from gate *E.* Reanalyzing stem *m* would result in an undetectable fault r s-a-1. Modifying the circuit would result in a direct connection from *m* to *u.* Since *f* is an undetectable fault and has no connection to primary output, gates *A* and *C* can be removed from the circuit. This results in the simplified

circuit as shown in Figure 2(b) as there are no more stems to analyze.

Results of applying this technique on selected benchmark circuits are shown in Table 2. We see that because of single stem analysis is adopted to remove redundant faults, redundant faults due to reconvergent gates with multiple stems are not detected.

4. Identification and Removal of Redundant Faults in Synchronous Sequential Circuits

Unlike combinational circuits, all untestable faults in sequential faults are not redundant. This makes identification of redundant faults in sequential circuits more complex. Sequential redundant faults were defined under different assumptions before the concept of partially detectable faults was introduced in [10]. In [10], it is observed that an undetectable fault, in a synchronous sequential circuit, may be observed at the output under certain initial conditions for any input sequence. Though the fault is undetectable, it is irredundant because it affects the circuit response under certain conditions. These faults are defined as partially detectable faults. However if the circuit operation is always started by applying a specific sequence, a fault may be redundant. Therefore, because of the relationship between the operation mode of the circuit and these redundant faults, such faults are called operationally redundant. Faults which are not partially testable are defined as redundant faults.

In [10], any restriction was not imposed on the time of observation of the response of the circuit while defining sequential redundancy. In [11], the concept of c-cycle redundancy was introduced to overcome this shortcoming. As the procedures used to detect and remove sequential redundancy are more complex, they are not discussed here. Multiple-stem analysis for sequential circuits has also been carried out and is described in detail in [12].

5. Conclusion

The importance of identifying and removing redundant faults, which increases the complexity of test generation for digital circuits, has been described in this paper. A method used to identify redundant faults and a method used to remove the detected faults and to simplify the circuit, has been discussed in detail. Few other methods that detect and remove these faults were cited. The complexity of finding redundant faults in synchronous sequential circuits was also presented. Since dynamic analysis detects and removes redundant faults during test generation, it can be faster than doing static analysis before test generation. Research is being carried out to invent new dynamic RID techniques or improve existing dynamic techniques particularly for sequential circuits.

References:

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

[2] Harihara, M. and Menon, P.R. "Identification of undetectable faults in combinational circuits" *Proc. ICCD*, pp: 290 – 293, 1989.

[3] Bryan, D., Brgles, F and Lisanke, R. "Redundancy identification and removal", *Proc. MCNC Workshop on Logic Synthesis,* 1989.

[4] Menon, P.R. and Ahuja, H. "Redundancy removal and simplification of combinational circuits" *IEEE VLSI Test Symposium*, pp: 68 – 273. 1992.

[5] Iyer, M.A. and Abramovici, M. "FIRE: a fault-independent combinational redundancy identification algorithm" *IEEE Trans. On VLSI Systems*, Vol: 4, No. 2, pp: 295 – 301, 1996.

[6] Frieman, A.D. "Fault detection in redundant circuits", *IEEE Trans. On Electronic Computers*, EC-16(6): 99-100, 1967.

[7] Ratiu, I.M., Vincentelli, S. and Pederson, D.O. "VICTOR: a fast VLSI testability analysis program", *Proc. ITC*, pp: 397-401, 1982.

[8] Abramovici, M. and Iyer, M.A. "One-Pass Redundancy Identification and Removal" *Proc. ITC* pp: 807, 1992.

[9] Menon, P.R., Ahuja, H. and Harihara, M. "Redundancy identification and removal in combinational circuits", *IEEE Trans. on CAD Design of Integrated Circuits and Systems*, Vol: 13, No. 5, pp: 646-651, 1994.

[10] Pomeranz, I. and Reddy, S.M. "Classification of faults in synchronous sequential circuits", *Proc. IEEE Trans. on Computes*, Vol: 42, No. 9, pp: 1066-1077, 1993.

[11] Iyer, M.A., Long, D.E. and Abramovici, M. "Identifying sequential redundancies without search", *Proc. DAC*, pp: 457-462, 1996.

[12] Peng, Q., Abramovici, M. and Savir, J. "MUST: multiple-stem analysis for identifying sequentially untestable faults", *Proc. ITC*, pp: 839-846, 2000.