# Multiple Faults: Modeling, Simulation and Test*

Yong Chang Kim†
University of Wisconsin-Madison
Madison, WI 53706 USA
*kimy@ece.wisc.edu*

Vishwani D. Agrawal
Agere Systems
Murray Hill, NJ 07974 USA
*va@agere.com*

Kewal K. Saluja
University of Wisconsin-Madison
Madison, WI 53706 USA
*saluja@engr.wisc.edu*

## Abstract

*We give an algorithm to model any given multiple stuck-at fault as a single stuck-at fault. The procedure requires insertion of at most $n + 3$ modeling gates, when the multiplicity of the targeted fault is $n$. We prove that the modeled circuit is functionally equivalent to the original circuit and the targeted multiple fault is equivalent to the modeled single stuck-at fault. The technique allows simulation and test generation for any arbitrary multiple fault in combinational or sequential circuits. We further demonstrate applications to bridging-fault modeling, diagnosis, circuit optimization, and testing of multiply-testable faults. The modeling technique has an additional application in a recently published combinational ATPG method for partial-scan circuits in which some lines are split, leading to a transformation of single stuck-at faults into multiple faults.*

## 1. Introduction

Even though there is little or no support for the argument that only single faults will occur in the nature, our test methodology continues to rely on the single-fault assumption. The reasons, as discussed in the literature [6], are: (a) a large percentage of multiple faults is covered by the single-fault tests [3, 13, 14], (b) specific design styles and test generation procesures guarantee the detection of multiple-faults if single faults were tested [5, 11, 17, 19], and perhaps (c) the number of multiple-faults is too large for an economical analysis. As a result, fault simulation and test generation tools are available only for single stuck-at faults.

It is true that the single-fault assumption has worked well in practice. However, there are situations that require multiple-fault analysis (both simulation and test generation) in some limited sense. For example, circuit optimization by single-fault redundancy removal can be further improved if we have the capability of finding redundant multiple-faults. Similarly, the resolution of a diagnostic procedure can be improved if we could find tests for multiple-faults constructed from the suspected fault set identified by the single-fault tests. We demonstrate these and several other applications in Section 3. We believe these are "open problems" of practical significance.

The main contribution of this paper is to provide a logic-level single stuck-at fault model for any arbitrary multiple stuck-at fault in a combinational or sequential circuit. This technique allows simulation and test generation for the modeled fault by any single-fault simulator or test generator. The single stuck-at fault has previously being used for modeling path delay faults [4].

Algorithms for generating tests for multiple stuck-at faults exist [1]. Programming of such algorithms will be useful if we were to change the test objective, because the technique considers all multiple-faults rather than targeting one or a few specified faults. For the types of applications we discuss in Section 3, we must deal with a few selected multiple-faults, and programming of a complex algorithm does not seem fruitful. Therefore, we propose a modeling technique that allows an effective use of the existing tools.

## 2. A New Model for Multiple Faults

Figure 1(a) shows four lines with inputs *a*, *b*, *c* and *e*, and the respective outputs *A*, *B*, *C* and *E*. A multiple stuck-at fault here consists of the first two lines stuck-at-1 and the others stuck-at-0. A multiple-fault involving any set of lines and any arbitrary fault conditions can be modeled in a similar manner. We will use the multiple stuck-at fault example of Figure 1(a) for explaining the modeling method.

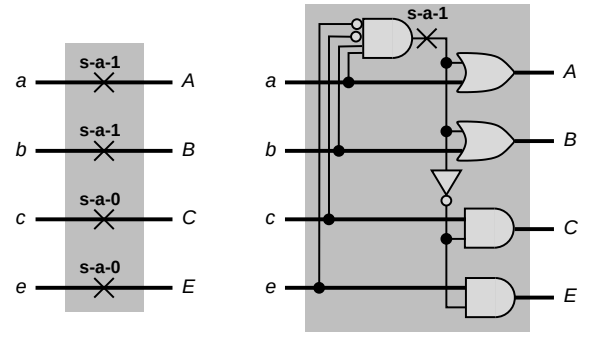
The model in Figure 1(b) contains a single stuck-at fault. It consists of two types of gates:

1. *In-line gates:* A two-input gate is inserted in each faulty line. The controlling input signal state for this gate is the same as the value at which the line is stuck. Thus, an AND (OR) gate is inserted in a line that is stuck-at-0 (1). When the fault on a line is not activated, the in-line gate forces the correct value on it.

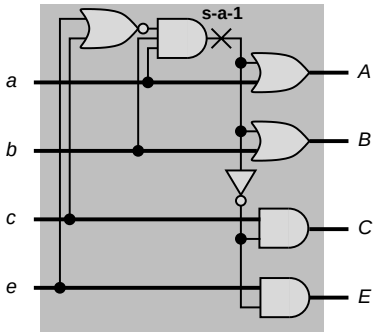
2. *Fault gate:* This is an $n$ input AND gate that feeds all in-line gates either directly if the in-line gate is OR type, or through an inverter if it is AND type. The inputs to the fault gate are derived directly from all s-a-1 lines and via inversions from all s-a-0 lines. A single s-a-1 fault

*This work was supported in part by the National Science Foundation grant MIP9714034.

†Summer Intern (2001) at Agere Systems, Murray Hill, NJ 07974.




(a) A multiple stuck-at fault. (b) An equivalent single stuck-at fault.




(c) An alternate implementation of the equivalent single stuck-at fault.

**Figure 1. A model for a multiple stuck-at fault.**

is modeled at the output of this gate. Thus, whenever faults are activated on any lines, the values of those lines activate the single fault in Figure 1(b) as $\overline{D}$. We use the symbols $D$ and $\overline{D}$ to represent the state of a line affected by fault. $D$ ($\overline{D}$) means that the fault-free (faulty) value is 1 (0) and faulty (fault-free) value is 0 (1) [6]. As a result, in-line gates inject $\overline{D}$ on all lines on which s-a-1 faults are activated and $D$ on lines on which s-a-0 faults are activated. A nine-valued algebra [18] may be required for the proper test generation for combinational and sequential circuits.

To prove that a model produced by the above construction is correct, we examine the example of Figure 1. We must prove two conditions:

- **Condition 1:** *Circuit equivalence*. Fault-free output functions must be identical for the original circuit and the model. For Figure 1(b), we have:

$$\begin{aligned} A &= a + a\,b\,\overline{c}\,\overline{e} = a \\ B &= b + a\,b\,\overline{c}\,\overline{e} = b \\ C &= c(\overline{ab\overline{c}\,\overline{e}}) = c(\overline{a}+\overline{b}+c+e) \\ &= c + c(\overline{a}+\overline{b}+e) = c \\ E &= e(\overline{ab\overline{c}\,\overline{e}}) = e(\overline{a}+\overline{b}+c+e) \\ &= e + e(\overline{a}+\overline{b}+c) = e \end{aligned}$$

which are same for the fault-free circuit in Figure 1(a). ■

- **Condition 2:** *Fault equivalence*. For the single s-a-1 fault in Figure 1(b) to be equivalent to the the multiple stuck-at fault in Figure 1(a), each of the four faulty functions should be identical [6]. Faulty functions for multiple-fault ($mf$) in Figure 1(a) and those for single-fault ($sf$) in Figure 1(b)) are:

$$\begin{aligned} A_{mf} &= 1 & A_{sf} &= a+1=1 \\ B_{mf} &= 1 & B_{sf} &= b+1=1 \\ C_{mf} &= 0 & C_{sf} &= c.0=0 \\ E_{mf} &= 0 & E_{sf} &= e.0=0 \end{aligned}$$

This proves the condition. ■

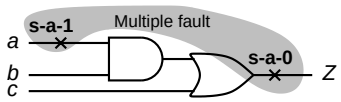
A multiple stuck-at fault of multiplicity $n$ can be modeled as a single stuck-at fault by using at most $n+3$ gates. Suppose out of $n$ faults, there are $(n-k)$ s-a-1 and $k$ s-a-0 faults, where $n \geq k$. Figure 1 (c) shows an alternative implementation using only $n+3$ additional gates. These include $n$ two-input *in-line* gates, one $(n-k+1)$-input *fault* AND gate, one $k$-input NOR gate and one inverter. When all n-component stuck-at faults are of the same type $n+1$ gates will be required.

It should be noted that the single-fault model can produce a feedback depending upon the locations of faults in the multiple-fault set. However, the model falls in the class of combinational circuits that contain "structural," but not "functional," feedback. For a discussion of such circuits, which have other applications, the reader may refer to the literature. A general theory for inverting large number of variables using the smallest number of inverters (with feedback) is discussed by Huffman [12]. Another application of a combinational circuit with feedback is described by Dietmeyer [10]. That application consists of a 1's complement adder circuit in which the end-around carry is realized by feeding the carry out from the most significant bit of the adder back to the carry-in input of the adder.

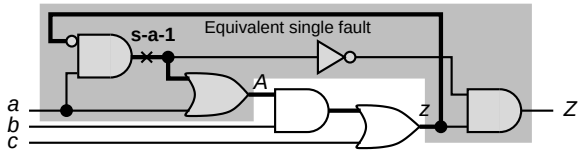
Figure 2(a) shows an example of a multiple-fault for which the single-fault model is given in Figure 2(b). The modeling portion of the circuit is enclosed in the shaded region and the structural feedback is shown with bold lines. We notice that the relationships $A = a$ and $Z = z$ are always "true," irrespective of whether or not the single stuck-at fault is present. The feedback path can never be sensitized and the circuit has no internal state. The two circuits in Figure 2 are functionally identical.

## 2.1. Multiple Faults for Sequential Circuits

The model we have shown in Figure 1 can also be used for the multiple faults in the combinational logic of any sequential circuit. The fault equivalence conditions 1 and 2 still hold regardless of sequential elements in the circuit. Any multiple stuck-at fault of multiplicity $n$ can still be modeled




(a) A multiple stuck-at fault.




(b) Equivalent single stuck-at fault with non-functional feedback.

**Figure 2. Illustration of non-functional feedback.**

with at most $n + 3$ gates. The modeling structure may create non-functional feedback within the combinational circuit logic but the seemingly acynchrounous behavior will never be activated.

## 3. Applications

There are several problems that require modeling of multiple stuck-at faults. These problems can be solved in today's VLSI design environment by the modeling technique of the previous section. The present work was motivated by the type of applications discussed in this section. We provide these with the hope that interested readers will find novel solutions to these and other similar problems.

### 3.1. Diagnosis

The purpose of diagnosis is to identify and locate a fault. Once a fault is identified, it can be repaired. Good diagnostic test should be able to exactly pinpoint the fault location, but the application of tests only tells us whether or not the system under test is faulty. Further analysis is required to determine the nature of the fault and to pinpoint the fault location. Most diagnostic procedures are based on single-fault tests and difficulties arise when we try to match the symptoms produced by "real" faults. Nevertheless, the *single-fault dictionary* approach [6, 7, 20] is a frequently used method. The following example illustrates how a multiple-fault capability can improve diagnosis.

Consider the circuit of Figure 3. We use a set of six vectors that detects all 16 *collapsed* single stuck-at faults. For an input order $(A, B, C, D)$, these vectors are $T_1 = (1111)$, $T_2 = (1101)$, $T_3 = (0001)$, $T_4 = (1010)$, $T_5 = (1110)$ and $T_6 = (0111)$. We use a subscript notation to denote a stuck-at fault. For example, a signal $A$ stuck-at-0 is denoted as $A_0$. In Figure 3, a signal name is either the label on the line or the label of the gate that produces it. To obtain a fault dictionary, we simulate the vectors without *fault dropping*. Table 1 gives the simulation result in a dictionary format [6]. The outcome of a test $T_i$ is expressed as a binary variable $t_i$, which is 1 if

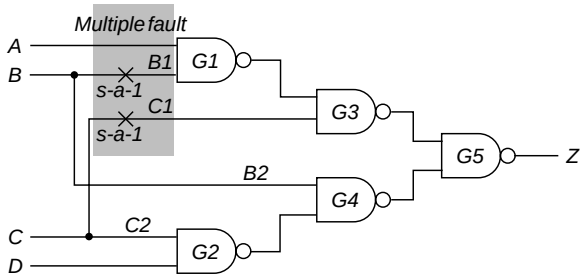


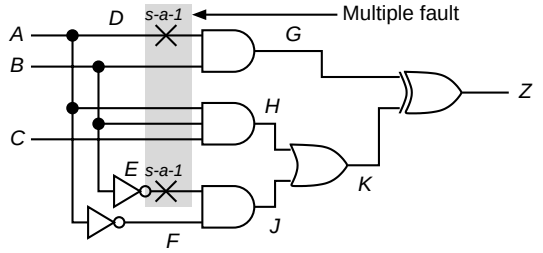

**Figure 3. An example circuit for diagnosis.**

**Table 1. Fault dictionary for diagnosis**

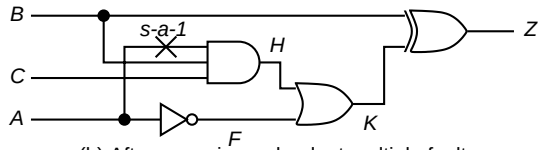
| Faults | Test syndromes | | | | | | Hamm. |
|---|---|---|---|---|---|---|---|
| | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | Dist. |
| *No fault* | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| $A_1$ | 0 | 0 | 0 | 0 | 0 | 1 | 3 |
| $D_1$ | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| $B1_1$ | 0 | 0 | 0 | 1 | 0 | 0 | **1** |
| $G3_1$ | 0 | 0 | 0 | 1 | 0 | 1 | 2 |
| $B_1$ | 0 | 0 | 1 | 0 | 0 | 0 | **1** |
| $C1_1$ | 0 | 0 | 1 | 0 | 0 | 0 | **1** |
| $B2_1$ | 0 | 0 | 1 | 0 | 0 | 0 | **1** |
| $C2_1$ | 0 | 1 | 0 | 0 | 0 | 0 | 3 |
| $G4_1$ | 0 | 1 | 0 | 0 | 1 | 0 | 4 |
| $Z_0$ | 0 | 1 | 0 | 1 | 1 | 1 | 4 |
| $C_1$ | 0 | 1 | 1 | 0 | 0 | 0 | 2 |
| $G1_1$ | 1 | 0 | 0 | 0 | 0 | 0 | 3 |
| $G2_1$ | 1 | 0 | 0 | 0 | 0 | 0 | 3 |
| $C_0$ | 1 | 0 | 0 | 1 | 0 | 0 | 2 |
| $Z_1$ | 1 | 0 | 1 | 0 | 0 | 0 | 2 |
| $B_0$ | 1 | 1 | 0 | 0 | 0 | 0 | 4 |
| *CUT* | 0 | 0 | 1 | 1 | 0 | 0 | – |
| $(B1_1, B_1)$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $\mathbf{(}B1_1, C1_1\mathbf{)}$ | 0 | 0 | 1 | 1 | 0 | 0 | **0** |
| $(B1_1, B2_1)$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $(B_1, C1_1)$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $(B_1, B2_1)$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $(C1_1, B2_1)$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $(B1_1, B_1, C1_1)$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $(B1_1, B_1, B2_1)$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $(B1_1, C1_1, B2_1)$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $(B_1, C1_1, B2_1)$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| $(B1_1, B_1, C1_1, B2_1)$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

a fault is detected by $T_i$ or 0, otherwise. Thus, each fault produces a sequence of six binary values, called *test syndrome*. Our *fault dictionary* contains a set of test syndromes associated with all single stuck-at faults. For *no fault*, the fault-free circuit has the test syndrome 000000.

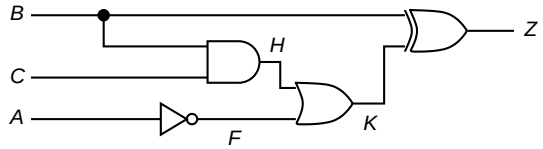
Suppose a circuit under test (CUT) contains a multiple fault, $(B1_1, C1_1)$, i.e., lines $B1$ and $C1$ stuck-at-1, simultaneously. The observed test syndrome is shown in Table 1 between two horizontal dividing lines. Since this test syndrome, 001100, does not match with the entries of single stuck-at faults in the dictionary, we use a distance approach [20]. The last column, *Hamm. Dist.*, in Table 1 shows the Hamming distance between the observed syndrome and that of each single stuck-at fault. Four faults with the smallest Hamming distance of 1 (shown in boldface) emerge as the suspected fault set, a set of most probable candidates, namely $B1_1$, $B_1$,




(a) A single-fault irredundant circuit with a redundant multiple fault.




(b) After removing redundant multiple fault, circuit has a redundant single fault.




(c) Irredundant circuit.

**Figure 4. Optimization by removing multiple-fault redundancy.**

$C1_1$ and $B2_1$. However, we are not able to reduce the suspected fault set any further just on the basis of the single-fault tests. We, therefore, use the multiple-fault model to derive the test syndromes assuming any subset of these faults may be present in the CUT. By simulating the circuit after injecting the multiple-faults as single stuck-at faults using the technique of Section 2, we generated the additional eleven test syndromes, one for each possible grouping. Ten out of eleven subsets have the same test syndrome of $001000$, showing a Hamming distance of 1 from CUT. Only the test syndrome for the multiple fault pair $B1_1$ and $C1_1$ has a Hamming distance of 0, indicating a "perfect match." Thus we conclude that $B1_1$ and $C1_1$ are presented as a multiple fault in the CUT.

In general, the accuracy of diagnosis will heavily depend on the tests used to create fault dictionary. Based on our experiments, we found that the best tests for the multiple fault diagnosis must satisfy following two conditions. First condition is that it must produce a non-zero test syndrome for CUT, differentiate CUT from the fault-free circuit. Second, it should produce a different test syndrome for CUT than the syndromes of single stuck-at faults. Further improvements may be made by generating new tests or using additional tests that target multiple-faults involving the suspected fault set. In other words, the diagnosis may be improved if a longer and/or a different set of test vectors is used.

### 3.2. Circuit Optimization

Logic circuits are optimized using combinational ATPG programs, which identify redundant single stuck-at faults.

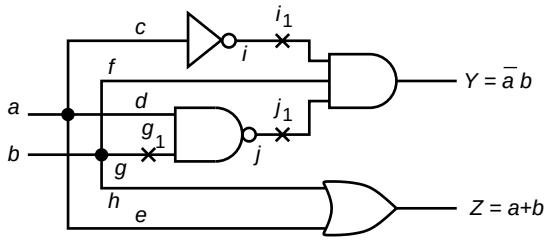



**Figure 5. Multiply-testable stuck-at faults.**

Only one redundant fault is removed at a time by fixing the signal at the site and deleting any unnecessary gates [6]. The process of one fault removal and combinational ATPG is repeated until the circuit becomes irredundant. The following example shows how such an irredundant circuit can be further optimized by considering multiple faults.

A single fault irredundant circuit is shown in Figure 4(a) [2]. All single stuck-at faults are detectable and hence a technique of redundancy removal, based on single faults, will not be able to optimize it. We find that a multiple-fault, $D$ s-a-1 and $E$ s-a-1 as shown in Figure 4(a), is redundant. Thus, setting lines $D$ and $E$ to 1, fixing all implied signals, and removing unnecessary logic, we get the circuit of Figure 4(b). Now, we find that this circuit has a new redundant single s-a-1 fault on the $A$ input of AND gate $H$. After removing that fault we get the minimal irredundant circuit of Figure 4(c). All three circuits are functionally identical. Notice that this reduction would not be possible without the analysis of multiple faults. Even with moderately large circuits, it will not be possible to model all multiple faults. It will be beneficial to develop analytical or heuristic procedures to determine which faults should be targeted for redundancy analysis. In circuits with several redundant single-faults, groups of such faults can be analyzed by the multiple-fault modeling method. Any multiple-fault found to be redundant can be removed, instead of removing one single faults at a time.

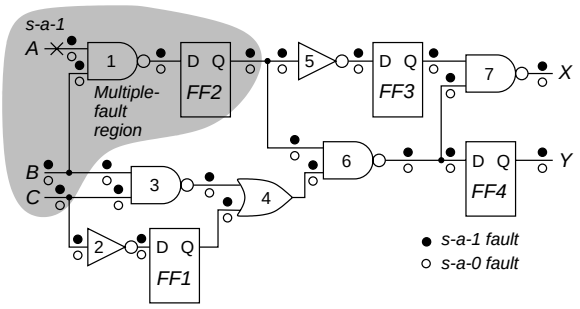
### 3.3. Multiply-Testable Stuck-at Faults

Sometimes a digital design may have untestable single stuck-at faults that cannot be removed. Typical reasons can be unused or partially used gates added during the standard cell mapping, or the logic added for speeding up critical paths. When untestable faults are present, it is useful to include tests for *multiply-testable faults*. A multiply-testable stuck-at fault is defined as a group of simultaneously occurring stuck-at faults, none of which is testable as a single fault [6].

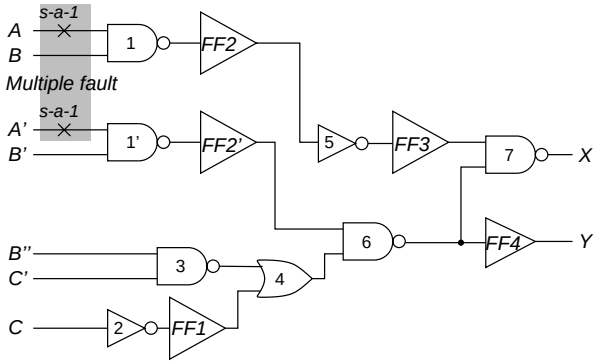
Figure 5 shows a circuit with three redundant single stuck-at faults: $g_1$, $i_1$ and $j_1$. All other single stuck-at faults are tested by three vectors: 00, 01 and 10. Although other vector sets are possible, these vectors were produced by a typical ATPG program. The three redundant faults form four multiple-faults whose tests are shown in Table 2. If for some practical reason, the circuit cannot be optimized or made

**Table 2. Multiple-fault tests for the circuit of Figure 5.**

| Multiple stuck-at fault | Test |
|---|---|
| $(g_1, i_1)$ | Redundant |
| $(g_1, j_1)$ | Redundant |
| $(i_1, j_1)$ | 11 |
| $(g_1, i_1, j_1)$ | 11 |




(a) An acyclic sequential circuit.




(b) A combinational model: fault A s-a-1 maps onto a multiple fault (A s-a-1, A' s-a-1).

**Figure 6. Combinational ATPG model of an acyclic circuit.**

completely single-fault testable, then by adding the vector 11 to the test set we can cover the "singly-untestable faults" when they occur as multiple-faults.

### 3.4. Combinational ATPG for Partial Scan

Since an acyclic circuit has a finite sequential depth, one can expand the time-frames and then generate tests for a multiple-fault using a combinational ATPG program. However, for large sequential depth this procedure can be expensive. Recent papers propose efficient procedures [15, 16]. In this case only some parts of the circuit are selectively duplicated so the combinational ATPG model is compact. Most faults map onto single stuck-at faults in the ATPG model. But some faults, usually a small percentage, map onto multiple-faults. The technique of Section 2 can be applied to model multiple-faults for detection by a single-fault ATPG program.

**Table 3. Combinational model ATPG results for partial-scan ISCAS '89 circuits.**

| Circuit name | Fault statistics | | ATPG CPU s on Sun Ultra 10 | | |
|---|---|---|---|---|---|
| | Total faults | % of MFs | Seq. method | Comb. method | Speed ratio |
| s382 | 399 | 0.0 | 0.18 | 0.03 | 4.2 |
| s400 | 424 | 0.0 | 0.13 | 0.04 | 2.3 |
| s444 | 474 | 0.0 | 0.15 | 0.05 | 1.9 |
| s641 | 467 | 0.8 | 0.34 | 0.08 | 3.0 |
| s713 | 581 | 0.9 | 1.02 | 0.34 | 2.0 |
| s953 | 1076 | 0.0 | 0.49 | 0.15 | 2.3 |
| s1196 | 1242 | 32.5 | 1.33 | 0.54 | 1.5 |
| s1238 | 1355 | 33.7 | 2.83 | 1.11 | 1.6 |
| s1423 | 1515 | 2.8 | 2.17 | 0.53 | 3.1 |
| s5378 | 4603 | 52.1 | 1268.00 | 23.30 | 53.4 |
| s9234 | 6927 | 9.2 | 425.63 | 85.68 | 4.0 |
| s13207 | 9815 | 31.1 | 1008.04 | 54.99 | 17.3 |
| s15850 | 11725 | 52.4 | 853.49 | 140.77 | 5.1 |
| s35932 | 39094 | 13.1 | 569.07 | 79.44 | 6.2 |
| s38417 | 31180 | 4.3 | 860.87 | 98.17 | 7.8 |
| s38584 | 36303 | 18.9 | 7293.11 | 239.65 | 29.4 |
| Average | 9199 | 15.7 | 767.93 | 45.30 | 16.0 |

Figure 6(a) shows an acyclic circuit and the uncollapsed faults on each line (• for stuck-at-1 fault and ◦ for stuck-at 0 fault), where the faults in the shaded area are the only ones that map onto multiple faults in the model. The generated combinational model is shown in Figure 6(b). As an example, the input $A$ stuck-at-1 fault in Figure 6(a) appears as a multiple-fault in the combinational ATPG model shown in Figure 6(b). Using the technique of Section 2, we can add modeling gates to the circuit of Figure 6(b) such that every single-fault of the sequential circuit is represented by a single-fault in the combinational circuit. As a result, a conventional combinational ATPG can be used.

The above procedure is only applicable to acyclic sequential circuits. However, a cycle-cutting method of partial scan [8] allows an acyclic test mode in any general sequential circuit. Table 3 shows the results of combinational test generation for *partial-scan* ISCAS '89 circuits. These circuits are sequential since they have one or more flip-flops remaining after cutting all cycles. The first column, *Circuit name*, shows the name of partial-scan ISCAS '89 benchmark circuit. Column *Total faults* shows the total number of collapsed faults in the sequential circuit and Column *% of MFs* shows the percentage of faults that was mapped as multiple-faults. Remaining columns show the test generation times for sequential and combinational methods. The Gentest [9] ATPG is used for the sequential method and TetraMax [21] is used for the combinational model method. Both results were obtained a Sun Ultra 10 workstation. The last column, *Speed ratio* shows test generation time speed up obtained by the combinational modeling method. The advantage of being able to use an optimized combinational ATPG program [21] is evident.

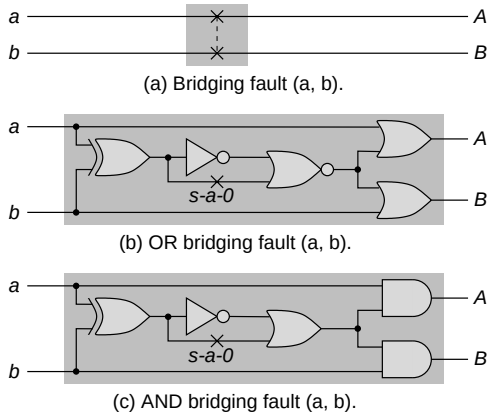



**Figure 7. OR and AND bridging fault models.**

### 3.5. Bridging-Fault Modeling

A *bridging fault* represents a short between a group of lines. The logic value of the shorted line may be modeled as 1-dominant *OR bridge* or 0-dominant *AND bridge*. The bridging fault cannot be modeled with a traditional single stuck-at fault model. However, one can model it as a single stuck-at fault using a variation of the multiple fault model proposed in Section 2. Figure 7 (a) shows a bridging fault (a, b), between line *a* and line *b*. Figures 7 (b) and (c) show how to model OR and AND bridging faults on *a* and *b* with a single stuck-at 0 fault.

## 4. Conclusion

The multiple-fault model we present is inefficient because it requires a large number of gates. It cannot be used if we must model all or many multiple faults. Clearly, it is not our intent to change the test methodology based on the single-fault assumption. Our motivation is derived from the type of applications given in Section 3, which require the analysis of a small number of multiple faults. The new model makes such applications possible through the use of the available tools. Extremely efficient single-fault ATPG programs are commercially available [21] for which moderate increases in the number of gates do not present any difficulty. Another strength of this model is its flexibility in transforming many problems into single-fault problems. We believe this will lead to future research on novel design and test methods.

## References

[1] E. M. Aboulhamid, Y. Karkouri, and E. Cerny, "On the Generation of Test Patterns for Multiple Faults," *J. Electronic Testing: Theory and Applic.*, vol. 4, no. 3, pp. 237–253, Aug. 1993.

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

[3] V. K. Agarwal and A. S. F. Fung, "Multiple Fault Testing of Large Circuits by Single Fault Test Sets," *IEEE Trans. Computers*, vol. C-30, no. 11, pp. 855–865, Nov. 1981.

[4] P. Agrawal, V. D. Agrawal, and S. C. Seth, "Generating Tests for Delay Faults in Nonscan Circuits," *IEEE Design & Test of Computers*, vol. 10, pp. 20–28, Mar. 1993.

[5] D. C. Bossen and S. J. Hong, "Cause-Effect Analysis for Multiple Fault Detection in Combinational Networks," *IEEE Trans. Computers*, vol. C-20, no. 11, pp. 1252–1257, Nov. 1971.

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

[7] H. Y. Chang, E. G. Manning, and G. Metze, *Fault Diagnosis of Digital Systems*. New York: Wiley-Interscience, 1970.

[8] K.-T. Cheng and V. D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedback," *IEEE Trans. Computers*, vol. 39, no. 4, pp. 544–548, Apr. 1990.

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

[10] D. L. Dietmeyer, *Logic Design of Digital Systems*. Boston: Allyn and Bacon, third edition, 1988.

[11] J. P. Hayes, "A NAND Model for Fault Diagnosis in Combinational Logic Networks," *IEEE Trans. Computers*, vol. C-20, no. 12, pp. 1496–1506, Dec. 1971.

[12] D. A. Huffman, "Combinational circuits with feedback," in A. Mukhopadhyay, editor, *Recent Deveolpments in Switching Theory*, New York: Academic press, 1971.

[13] J. L. A. Hughes and E. J. McCluskey, "Multiple Stuck-at Fault Coverage of Single Stuck-at Fault Test Sets," in *Proc. Int. Test Conf.*, Sept. 1986, pp. 368–374.

[14] J. Jacob and N. N. Biswas, "GTBD Faults and Lower Bounds on Multiple Fault Coverage of Single Fault Test Sets," in *Proc. Int. Test Conf.*, Sept. 1988, pp. 849–855.

[15] Y. C. Kim, V. D. Agrawal, and K. K. Saluja, "Combinational Test Generation for Acyclic Sequential Circuits using a Balanced ATPG Model," in *Proc. 14th Int. Conf. on VLSI Design*, Jan. 2001, pp. 143–148.

[16] Y. C. Kim, V. D. Agrawal, and K. K. Saluja, "Combinational Test Generation for Various Clases of Acyclic Sequential Circuits," in *Proc. Int. Test Conf.*, Oct. 2001.

[17] I. Kohavi and Z. Kohavi, "Detection of Multiple Faults in Combinational Logic Networks," *IEEE Trans. Computers*, vol. C-21, no. 6, pp. 556–568, June 1972.

[18] P. Muth, "A Nine-Value Circuit Model for Test Generation," *IEEE Trans. Computers*, vol. C-25, no. 6, pp. 630–636, June 1976.

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

[20] W. R. Simpson and J. W. Sheppard, *System Test and Diagnosis*. Boston: Kluwer Academic Publishers, 1994.

[21] Synopsys, Inc., 700 East Middlefield Rd., Mountain View, CA 94043, *TetraMAX ATPG User Guide*, v2000.11 edition, November 2000. Document Order Number: 37043-000 TBD.