terms in $F(\bar{F})$. Thus $S_{F_{\min}}$ will be the decimal sum of the smallest two minterms of F, and $S_{F_{\max}}$ that of the largest 2. Then

$$S_{F_{\min}} = 62 + 63 = 125 \qquad S_{F_{\max}} = 126 + 127 = 253$$

$$S_{\bar{F}_{\min}} = 0 + 1 = 1 \qquad S_{\bar{F}_{\max}} = 123 + 125 = 248$$

$$S_{G_{\min}} = 111 + 119 = 230 \qquad S_{G_{\max}} = 126 + 127 = 253$$

$$S_{\bar{G}_{\min}} = 0 + 1 = 1 \qquad S_{\bar{G}_{\max}} = 121 + 122 = 243.$$

If we use F we are obliged to generate all the S-sequences whose decimal sums lie in the range (125, 248), whereas if we use G we are required to generate only those S-sequences whose decimal sums lie in the range (230, 243). The second range being smaller, let us choose G for testing 2-asummability of F for more-than-five-1 2-sums.

Using notations of [8], n_0 can take the values $n_0 = 0, 1$. Using the algorithm from [8], we find that there is no S-sequence with the given range of n_0 whose decimal sum lies within the trange (230, 243). Hence, we conclude that F is 2-asummable.

Though a smaller range may not guarantee a smaller number of S-sequences to be generated, it has been found that in the canonical form of F less labor is required for testing.

IV. Conclusions

In the earlier sections a fast algorithm for testing 2-asummability of switching functions has been developed, which first tests 2-monotonicity. The 2-monotonicity is tested by a novel algorithm developed in Section II. The relationship between 2-monotonicity and 2-asummability investigated in Section III makes the 2-asummability algorithm of the authors [8] very fast, since it has been discovered there that 2-monotonicity is equivalent to testing 2-asummability for upto-five-1 2-sums. 2-asummability testing is normally done to ascertain linear separability of switching functions of upto eight variables. Empirically it has been found that Algorithm 1 presented above is superior to other techniques for testing 2-asummability for $n \leq 8$.

References

[1] R. O. Fontao, "A graphical method for checking 2-summability," *IEEE Trans. Comput.*, vol. C-20, pp. 461–464, Apr. 1971.
[2] S. Gosh, S. Bandyopadhyaya, S. K. Mitra, and A. K. Choudhury, "Simple methods for testing 2-summability of Boolean functions and isobaricity of threshold functions," *IEEE Trans. Comput.*, vol. C-21, pp. 503–507, May 1972.
[3] J. E. Hopcroft and R. L. Mattson, "Synthesis of minimal threshold logic networks," *IEEE Trans. Comput.*, vol. EC-14, pp. 552–560, Aug. 1965.
[4] S. Muroga, *Threshold Logic and Its Applications*. New York: Wiley-Interscience, 1971.
[5] S. Muroga, I. Toda, and S. Takasu, "Theory of majority decision elements," *J. Franklin Inst.*, vol. 271, pp. 376–418, May 1961.
[6] M. C. Paull and E. J. McCluskey, Jr., "Boolean functions realizable with single threshold devices," *Proc. IRE*, vol. 48, pp. 1335–1337, July 1960.
[7] A. K. Sarje, "Studies on the structural, asummability and realization aspects of threshold functions," Ph.D. dissertation, Indian Institute of Science, Bangalore, India, 1975.
[8] A. K. Sarje and N. N. Biswas, "An algorithm for testing 2-asummability of Boolean functions," *IEEE Trans. Comput.*, vol. C-26, pp. 1049–1053, Oct. 1977.
[9] P. K. Sinha Roy, "A slide rule device for checking 2-summability," *IEEE Trans. Comput.*, vol. C-17, pp. 279–283, Mar. 1968.
[10] Sureshchander, "An algorithm for testing asummability of Boolean functions," *IEEE Trans. Comput.*, vol. C-23, pp. 188–191, Feb. 1974.
[11] R. O. Winder, "Threshold logic," Ph.D. dissertation, Dep. of Mathematics, Princeton University, Princeton, NJ, 1962.

On Necessary and Sufficient Conditions for Multiple Fault Undetectability

JAMES E. SMITH

Abstract—**This correspondence states necessary and sufficient conditions for a multiple stuck-at fault in a combinational network to be undetected by a test set. The conditions are given in terms of fault masking relationships. It is shown that several other statements on this subject which have appeared in the literature are invalid.**

Index Terms—**Fault detection, fault masking, multiple faults, multiple fault test sets.**

I. Introduction

Some algorithms for generating test sets for multiple stuck-at faults in combinational networks begin by finding a test set for some critical set of single faults (possibly all single faults). Then, by examining fault masking relations, tests are added to detect any undetected multiple faults. Algorithms found in [1], [2] fit into this category. Unfortunately, there is some misunderstanding as to what masking conditions are necessary and sufficient for a multiple fault to be undetectable. This correspondence presents a set of necessary and sufficient conditions in terms of fault masking relations.

We consider single-output combinational circuits made of AND, OR, NAND, NOR, and NOT gates. We assume that a *fault* can be modeled as gate input or output lines stuck-at logical values. A *single fault* consists of only one stuck line while a *multiple fault* may consist of any number of stuck lines. A multiple fault F_j is represented as the set of its single fault components. That is, $F_j = \{f_1, f_2, \cdots, f_k\}$ where f_i represents a single stuck line.

A test input vector X_t designed to detect a fault F_j may fail to detect F_j in the presence of another fault. We say that a fault F_i *masks* a fault F_j *under test* X_t if F_j is detected by X_t but $F_i \cup F_j$ is not.

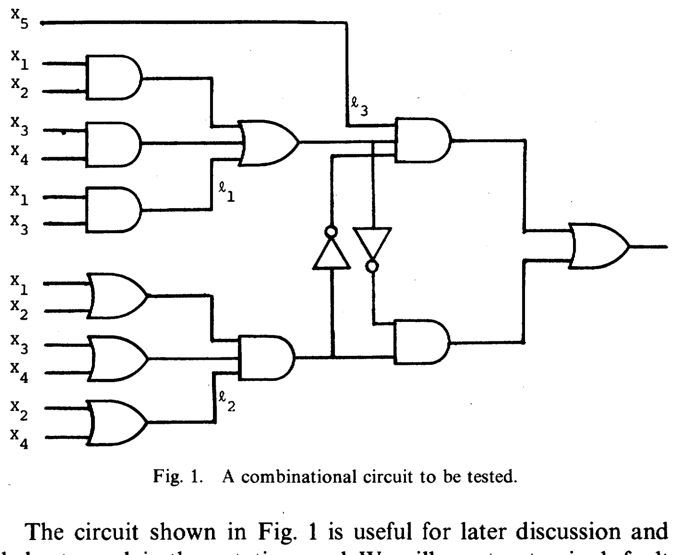

Fig. 1. A combinational circuit to be tested.

The circuit shown in Fig. 1 is useful for later discussion and helps to explain the notation used. We will construct a single fault

Manuscript received November 17, 1976. This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract DAAB-07-72-C-0259.

The author was with the Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801. He is now with the University of Wisconsin-Madison, Madison, WI 53706.

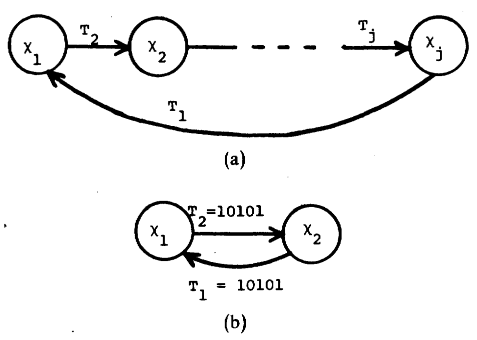

Fig. 2. (a) Fig. 10 from [3]. (b) Masking relations for the example circuit.

test set T_s for the circuit. The only test for $F_1 = \{l_1/0\}$ is $X_a = 10101$, hence $X_a \in T_s$. The only test for $F_2 = \{l_2/1\}$ is also X_a. It happens that F_1 masks F_2 under X_a and F_2 masks F_1 under X_a. The test $X_b = 10100$ detects $F_3 = \{l_3/1\}$, and we place X_b in T_s. The test X_b is also the only test for $F_4 = F_1 \cup F_2 = \{l_1/0, l_2/1\}$. Tests for the remaining single faults can be found and added to T_s in any fashion.

II. Discussion and Theorem

We are now ready to explore masking relationships that lead to undetectable multiple faults under a test set. We begin by discussing relationships given in the literature. These relationships are examined in terms of the circuit and test set T_s just given.

In the last paragraph of [1, p. 855] it is stated that "A multiple fault $F = \{f_1, f_2, \cdots, f_n\}$ is not detected by a single fault test set T_s if every f_k, $k = 1, 2, \cdots, n$ is masked, under T_s, by some subfault of F." The example given earlier shows this condition to be insufficient because F_4 is detected by T_s while every single fault belonging to F_4 is masked under T_s by a subfault of F_4.

The converse of the above statement is correct, however. This is proved in [5] for any test set T, not just single fault test sets.

In [2] masking relations are represented by a directed graph, and a different notation than ours is used. A fault labels each node of the graph, and an arc goes from node χ_i to node χ_j with label T_k if fault χ_i masks fault χ_j under test T_k. Definition 8 in [2] states, "Let $\chi = \{\chi_1, \chi_2, \cdots, \chi_j\}$ be a set of faults. χ forms a masking loop under test set T if both conditions 1) and 2) are satisfied.

1) By relabeling the faults in set χ and tests in test set T, the following masking relations can be established by Fig. 10 [our Fig. 2(a)].

2) For any fault (χ_i) in (χ) and for any test T_j ($T_j \in T$) which detects χ_i, there exists a set of faults containing χ_i and a subset of tests in T, which exhibit a set of masking relations as in 1."

Reference [2] then goes on, "Definition 8 says that the multiple fault (χ) is undetectable under the presence of test set T." Again, the example we have given shows this is not the case if $\chi = \{\chi_1, \chi_2\}$, $\chi_1 = \{l_1/0\}$, $\chi_2 = \{l_2/1\}$, and $T = T_s$. Fig. 2(b) shows the masking relations for our example using the notation of [2]. Although [2] discusses all-NAND networks, the circuit in Fig. 1 can be transformed to an all-NAND network, and faults equivalent to χ_1, χ_2, and χ can be found which have the same masking relationships under T_s as those given above.

Another incorrect statement on sufficient conditions for multiple fault undetectability appears in [3] and is pointed out in [4].

Necessary and sufficient conditions for multiple fault undetectability are now given and proved.

Theorem: A fault F in the circuit C is not detected by a test set T if and only if for each nonempty $F_i \subseteq F$ and for each $X_t \in T$ that detects F_i there is some nonempty $F_j \subseteq F$ such that F_j masks F_i under X_t.

Proof: (*If*) If we set $F_i = F$ and if the condition of the theorem is satisfied, then for each $X_t \in T$ that detects F there is some $F_j \subseteq F$ that masks F under X_t. Such a masking fault must always be present when F is present. If F is masked it is by definition not detected. This leads us to the apparently contradictory statement that if F is detected by some test in T it is not detected by the test. This statement can only be true if F is not detected by any test in T.

(*Only if*) We prove the contrapositive. Say there is a nonempty $F_i \subseteq F$ where $X_t \in T$ detects F_i, but there is no nonempty $F_j \subseteq F$ that masks F_i under X_t. Then any faults that mask F_i under X_t must contain an element not in F. Hence, if F alone is present, F_i must be detected by X_t since no masking fault can be present. Hence, X_t also detects F. Q.E.D.

This theorem and the earlier example imply that for an algorithm of the type mentioned above to determine whether a multiple fault is undetected, it may be necessary to determine whether all the subfaults of the fault in question are masked. Some of the incorrect statements given earlier represent attempts to avoid this time-consuming process by examining only the masking of particular subfaults (e.g., single faults). Nevertheless, it should be pointed out that the algorithms themselves are only incorrect in that they may assume a multiple fault is not detected when it actually is. If this happens, another test may be generated for the fault. The net effect is that the final test set will still detect all multiple faults, but it may contain unnecessary tests.

References

[1] M. Fridrich and W. A. Davis, "Minimal fault tests for combinational networks," *IEEE Trans. Comput.*, vol. C-23, pp. 850–859, Aug. 1974.
[2] F. J. O. Dias, "Fault masking in combinational logic circuits," *IEEE Trans. Comput.*, vol. C-24, pp. 476–482, May 1975.
[3] J. W. Gault *et al.*, "Multiple fault detection in combinational networks," *IEEE Trans. Comput.*, vol. C-21, pp. 31–36, Jan. 1972.
[4] B. Bose, "Comments on 'Multiple fault detection in combinational networks,'" this issue, p. 804.
[5] D. T. Wang, "Properties of faults and criticalities of values under tests for combinational networks," *IEEE Trans. Comput.*, vol. C-24, pp. 746–750, July 1975.

Comments on "A Note on Synchronizer or Interlock Maloperation"

THOMAS J. CHANEY

***Abstract*—E. G. Wormald's note[1] proposes a way to prevent metastable action in synchronizers. Experimental results from testing his suggested circuits show that his solution *does not work*. A reference to a general proof that synchronizers *must* have a region of metastable action is given.**

***Index Terms*—Arbiter, asynchronous interactions, flip-flop metastability, glitch, interrupt failure, synchronizer failure.**

Wormald[1] assumed, as have several others [1], that a Schmitt trigger circuit will "standardize" any input waveform and thus can be used as part of a flip-flop circuit to provide a circuit which has

Manuscript received May 15, 1977. This work was supported by the Division of Research Resources of the National Institutes of Health under Grant RR-00396.

The author is with the Computer Systems Laboratory, Washington University, St. Louis, MO 63110.