# Random Pattern Testability

JACOB SAVIR, MEMBER, IEEE, GARY S. DITLOW, MEMBER, IEEE, AND
PAUL H. BARDELL, SENIOR MEMBER, IEEE

*Abstract*—**A major problem in self testing with random inputs is verification of the test quality, i.e., the computation of the fault coverage. The brute-force approach of using full-fault simulation does not seem attractive because of the logic structure volume, and the CPU time encountered. A new approach is therefore necessary. This paper describes a new analytical method of computing the fault coverage that is fast compared with simulation. If the fault coverage falls below a certain threshold, it is possible to identify the "random-pattern-resistant" faults, modify the logic to make them easy to detect, and thus, increase the fault coverage of the random test.**

*Index Terms*—**Detection probability, fault coverage, random patterns, self test, signal probability.**

## INTRODUCTION

THE difficulty of test generation has been known for some time [1], [2]. Although the process has been shown to be, in general, NP complete, Goel's [3] experience is that computation time is close to an $N^2$ curve, where $N$ is the number of gates in the circuit. Since this kind of time complexity fails to produce test patterns in a reasonable amount of time in a VLSI environment, new approaches to the testing problem have been investigated.

With the introduction of VLSI it has become apparent that some investment in chip area to ease the testing problem is more than worthwhile. Design for testability in reference to self test (or built-in test) has become a prime issue. Simply speaking, the idea was to have the chip (card, subassembly, etc.) test itself. Two basic self-test strategies have been proposed, *exhaustive self test* and *random self test*. The class of exhaustive self test [4]–[8] proposes to partition the logic into small enough pieces that exhaustive application of inputs could be practical. Some sort of data compression is used to store the expected responses on chip. The problem with this approach is that good partitioning programs are virtually nonexistent, and unless this problem is successfully solved this solution will remain an academic exercise. The class of random self test uses random input stimuli rather than an exhaustive set. As a result, partitioning is avoided. The number of random inputs applied during test is closely related to the affordable test time and can be roughly ten million for high performance circuits. Several random self-test architectures have been proposed in the literature [9]–[12]. Basically speaking, all these versions use a linear feedback shift register (LFSR) to generate pseudo-random inputs that are fed to the circuit, and a multiple-input signature register (MISR) [10] to compress the circuit responses. With level-sensitive scan design (LSSD) principles followed [13], it is easy to add some gates to have the shift register latches (SRL's) function as an LFSR or an MISR in self-test mode. A second implementation of the same basic idea could be to leave the original SRL's intact, add an LFSR as input stimulus, a MISR as an output response compressor, and use the scan-in/scan-out ports to transport data from the input LFSR to the network's SRL chain, and from the output SRL chain to the MISR by SRL string loads [9].

Although the random self test has removed the partitioning burden, it has introduced the problem of test quality verification. In an exhaustive self test this is not an issue since proper testable design can guarantee that all single faults be covered [4]. However, application of a random pattern test, which is far from being exhaustive, may require a quality verification, especially when high fault coverages are required. For a self test to replace the traditional test generation/fault simulation process, it is necessary that its resulting fault coverage be virtually equivalent. Since simulation of long random patterns against large logic structures does not seem practical, we have to seek an analytical solution.

The random pattern testability problem can be spelled out simply as follows. Given a combinational circuit that has to be tested with $N$ random patterns, either prove that 100-$\epsilon$ percent of the population's single stuck-faults would be detected with probability $1 - \delta$, or modify the logic to satisfy this. Typically, we would like to have $\epsilon < 2$, $\delta < 0.001$.

The object of this paper is to show a cost-effective method to solve this problem. An attempt is made to identify all the random pattern-resistant faults, namely, those faults whose detection probability falls below a given threshold. Then, extra logic would be used to increase the probability of detecting those "hard faults" by making them more susceptible to random inputs. The value of this threshold is directly related to the affordable test time and is roughly $10^{-6}$ for a random pattern test of ten million patterns. Details of this issue will be discussed in a future paper.

To achieve this goal we show that there is a relationship between signal probability [14], [15] and detection probability. In fact, a signal probability program can serve as an engine to compute detection probabilities. Since the signal probability algorithms reported to date [14]–[18] suffer from an exponential storage problem it was necessary to develop a new algorithm that trades off accuracy in order to decrease the space complexity. The cutting algorithm reported in this paper computes signal probability bounds, rather than the exact

Manuscript received November 8, 1982; revised August 15, 1983.

J. Savir is with the IBM Data Systems Division, 265/415, Poughkeepsie, NY 12602, on leave from the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

G. S. Ditlow is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

P. H. Bardell is with the IBM Data Systems Division, 265/938, Poughkeepsie, NY 12602.

0018-9340/84/0100-0079$01.00 

figure, with the advantage that both the time and space complexities are practically linear. By using the cutting algorithm it is possible to compute a lower bound of the detection probability. Then, if this lower bound is larger than the threshold it may be marked off as an "easy fault." Otherwise it will be marked as a "hard fault," whose detection probability would have to be upgraded by logic modification.

The cutting algorithm takes advantage of the fact that exact computation of signal probabilities in tree networks can be done in linear time and space complexities. Thus, in order to compute signal probabilities in a general-type combinational circuit, the cutting algorithm "cuts" fan-out branches to turn the circuit into a tree. In the paper we show which probability bounds to assign to the cutpoints so that the signal probability computation in the tree network is compatible with that in the original network. All that is left is to propagate those bounds, on the equivalent tree, to the rest of the lines.

In following sections we describe 1) the relationship between signal probability and detection probability, 2) the cutting algorithm, which is central to our random pattern testability analysis, 3) some heuristics that can be used with the cutting algorithm to make it efficient, 4) the algorithm of computing lower bounds for the detection probability, which is based on the cutting algorithm, 5) some thoughts on how to correct hard faults once they are identified, and 6) some cutting algorithm applications to other disciplines. Finally, we summarize the main ideas and draw conclusions. The fault analysis is restricted to single stuck-at faults.

## Relationship Between Signal and Detection Probabilities

The signal probability of line $w$, $p_s(w)$, is the probability that a randomly selected input vector will yield $w = 1$. The detection probability of a fault $w/i$ (read line $w$ stuck-at-$i$ $\epsilon$ {0, 1}), $p_d(w/i)$, is the probability that a randomly selected input vector will cause at least one of the primary outputs to assume a wrong value. For simplicity assume the circuit has a single output, $F$.

Suppose first that there exists only one propagation path from $w$ to $F$, as shown in Fig. 1. Assume that the fault is $w/0$. Detection of this fault requires having an input vector that will force both inputs of gate $G_1$ to 1; the input of $G_2$ that does not lie on the propagation path should be forced to a 0; the input of $G_k$ that does not lie on the propagation path should be forced to 0, etc. This set of conditions can be put together by introducing an auxiliary gate $G^0$, whose output $F^0 = 1$ if and only if the previous set of sensitizing conditions is satisfied. Thus, a line that has to be forced to a zero (according to the sensitizing pattern) should be complemented when connected to $G^0$; and a line that has to be forced to a one should be left uninverted. With this arrangement we have turned the detection probability to a signal probability problem:

$$p_d(w/i) = p_s(F^0). \tag{1}$$

Fig. 1 displays the special case where $i = 0$. Suppose now that there are $j$ paths from line $w$ to $F$. There may be, in general, $2^j - 1$ propagation patterns between $w$ and $F$, counting all single and multiple paths.

Suppose further that an auxiliary gate, $G^q$, is introduced whose output $F^q = 1$ if and only if the fault on $w$ propagates along the $q$th path (*single path*) to the output $F$. Then, obviously,

$$p_d(w/i) \geq p_s(F^q). \tag{2}$$

Thus, this signal probability is a lower bound to the detection probability of the fault $w/i$. The cutting algorithm presented here rests on this relationship, namely, that it is possible to compute lower bounds on the detection probabilities by computing signal probabilities of auxiliary gates.

*Definition 1:* The *prime faults* are the faults on the input lines and on the origin of fan-out branches.

In Fig. 2, the origin of the fan-out branches are lines $\{w_1, w_2, \cdots, w_m\}$.

*Definition 2:* A fault, $w/i$, is *random-pattern testable* if $p_d(w/i) > p_{dt}$, where $p_{dt}$ is a given threshold.

The following well-known theorem in deterministic testing is also true in random testing [19].

*Theorem 1:* A circuit is random-pattern testable if all its prime faults are.

*Proof:* Since all the prime faults are random-pattern testable, then their detection probability is larger than the threshold, $p_{dt}$. However, since the prime faults dominate all the other faults in the circuit, the detection probability of all the other faults is at least $p_{dt}$. Q.E.D.

Theorem 1 will be our vehicle to prove random-pattern testability of any given circuit.

It is important to note that since our random-pattern testability analysis rests on examining single path propagation of faults, it may end up giving a zero lower bound for detection of a fault that can only be propagated through multiple paths. In these cases the resulting lower bound will be too pessimistic. To solve this problem we have two options: either simulate the circuit with this fault to get a more realistic figure, or introduce some extra hardware to force an acceptable single-path propagation.

## The Cutting Algorithm

### *The Full-Range Cutting Algorithm*

The cutting algorithm is the work-horse of the signal probability estimation engine. Its objective is to turn the combinational network into a tree, by cutting reconvergent fanout branches, and inserting equivalent bounds at the cut points, which will guarantee that all the signal probability bounds computed on this tree will enclose the true values. The advantage of doing this is the reduction of the computational complexity (time and space); the disadvantage is that the output is not an exact figure but just a bound.

In the full-range cutting algorithm we assign a signal probability range of [0, 1] to all the cut points, and propagate the bounds to all the other lines of the circuit by using tree formulas. Notice that in order to cut an $n$-way fan-out it is only necessary to cut $n - 1$ of its branches to turn it into a tree. Therefore, the line not cut receives the signal probability of its immediate ancestor. Fig. 3 illustrates this point.

Intuitively, at least, it should be clear that this assignment

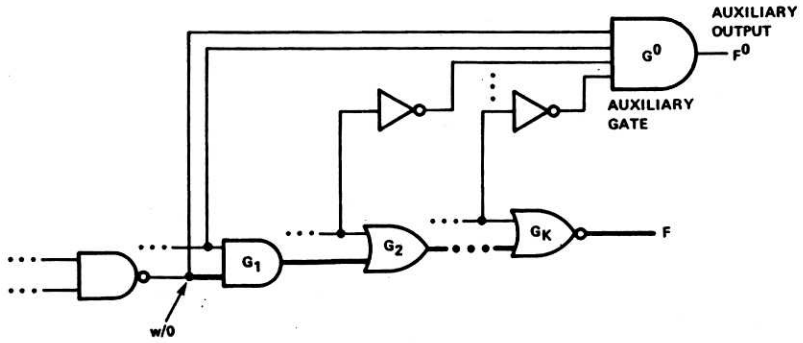


Fig. 1. An example for which there is only one propagation path from the site of the fault to a primary output.

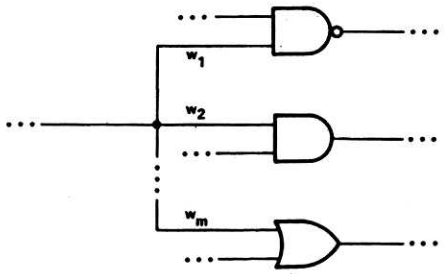


Fig. 2. An illustration of the origin of fan-out branches, $\{w_1, w_2, \cdots, w_m\}$.

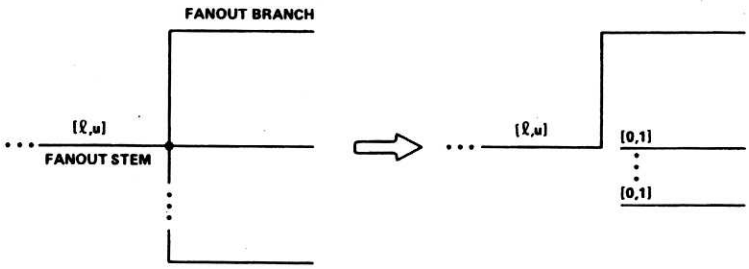


Fig. 3. The full-range cutting process performed on an $n$-way reconvergent fan-out point (reconverging gate not shown).

always works, because this covers the full range of the probability spectrum. Before we present a formal proof we define some terms, describe the algorithm, and show an example.

*Definition 3:* The *cone of influence* of a line $w$, $c(w)$, is the logic that feeds, directly or indirectly, this line.

*Definition 4:* A *tree line* is a line, $w$, for which $c(w)$ is a tree.

The signal probability of all the tree lines can be computed without the cutting algorithm. These values will naturally be exact. The cutting algorithm will enable one to compute signal probability bounds for all the nontree lines as follows.

*Procedure 1:*

*Step 1:* Assign signal probability of $\frac{1}{2}$ to all input lines.

*Step 2:* Compute the signal probability for all the tree-lines.

*Step 3:* Turn the circuit into a tree by cutting reconvergent fanout branches according to Fig. 3.

*Step 4:* Propagate the signal probability bounds to all the nontree lines by using tree formulas (Fig. 4).

Fig. 4 displays the formulas of propagating signal probability bounds through an INVERTER, AND, and OR gates. The formulas for NAND and NOR can be easily derived from Fig. 4 by noting that NAND is AND-NOT, and that NOR is OR-NOT.

*Example 1:* Consider the circuit of Fig. 5. In (a) we compute the signal probabilities of all the tree lines. In (b) we compute the signal probabilities of the nontree lines. The complete set of signal probabilities is shown in (c). The only nontree lines in Fig. 5 are lines $\{w_1, w_2\}$. The Boolean functions realized by these lines are

$$w_1 = x_1(\bar{x}_2 + x_3x_4) + x_5(\bar{x}_3 + \bar{x}_4), \text{ and}$$

$$w_2 = \bar{x}_5[\bar{x}_1 + x_2(\bar{x}_3 + \bar{x}_4)] + \bar{x}_1x_3x_4 + x_6(\bar{x}_3 + \bar{x}_4).$$

Note that the exact signal probabilities for these lines are within the computed bounds.

$$p_s(w_1) = 19/32 \in [1/4, 3/4],$$

$$p_s(w_2) = 41/64 \in [17/32, 27/32].$$

*Theorem 2:* The full-range cutting algorithm computes true bounds.

*Proof:* There is a one-to-one correspondence between set operations and logic gates. AND corresponds to intersection; OR to union; NOT to complementation, etc. Let the universe

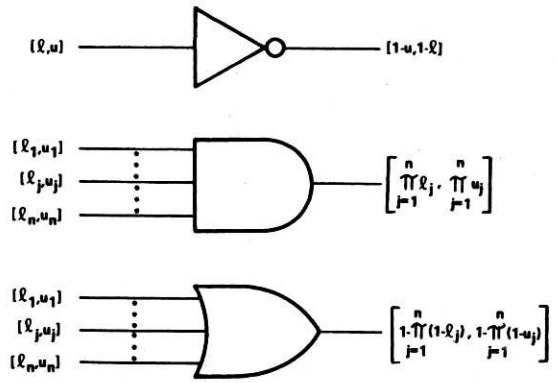


Fig. 4. Tree formulas for propagating signal probability bounds.

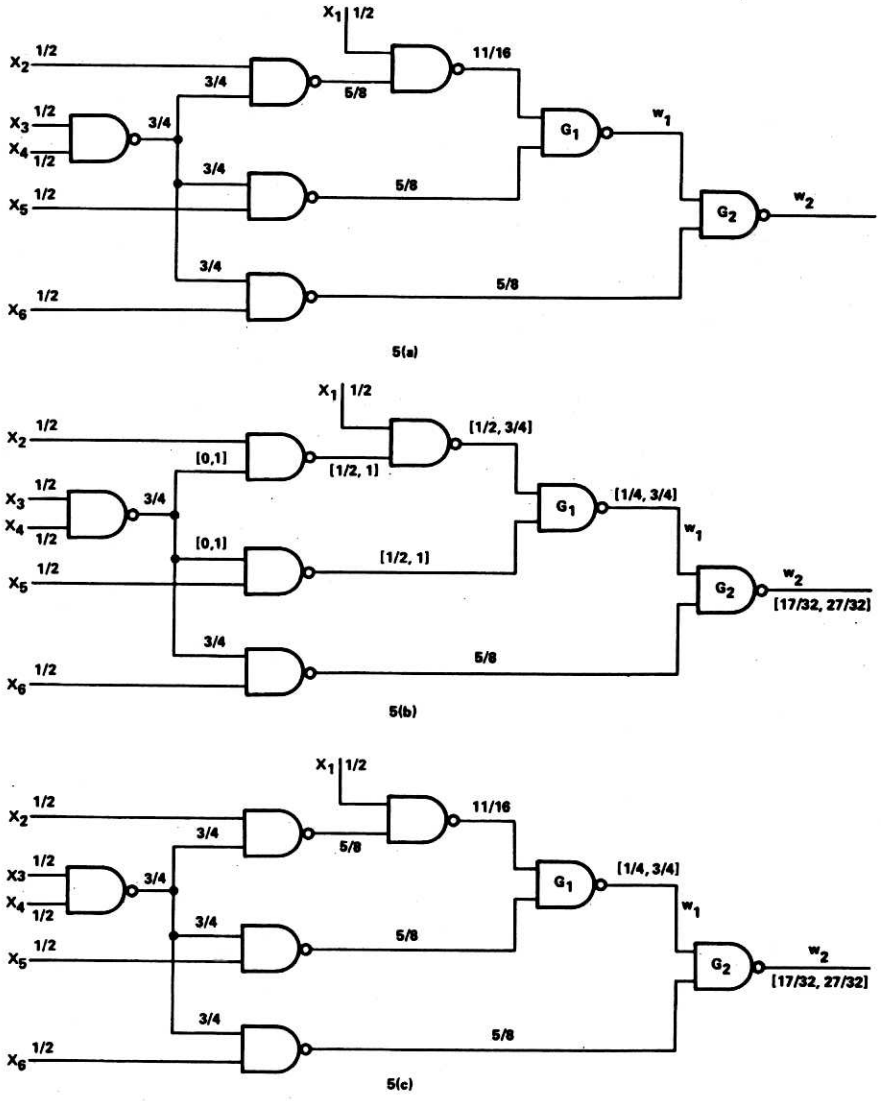


Fig. 5. (a) Signal probabilities for tree-lines. (b) Signal probability for nontree lines. (c) The final set of signal probabilities.

be the set of all possible input vectors. Assume that we assign to each input line the subset of input vectors for which the line assumes the value one (called the on-set) and that we perform the set operations that correspond to the logic gates that the circuit utilizes.

By performing those operations, the resulting set attached to each line in the circuit will be the subset of all input vectors that force that line to have the value one. Thus, the normalized cardinality of each such set is nothing but the signal probability of that line (normalized with respect to the cardinality of the universe).

A cutting of a fan-out branch corresponds to a replacement of the on-set of that line by two sets: the null set and the set universe. Assume that Procedure 1 is followed to completion with the associated sets. Then all the set operations done on the modified circuit (which is a tree) will compute subsets or

supersets of the on-sets of all the lines in the circuit. In other words, if the on-set of line $w_i$ is $OS(w_i)$, then the cutting algorithm will compute the set $OS^*(w_i)$ and $OS^{**}(w_i)$ such that $OS^*(w_i) \subseteq OS(w_i) \subseteq OS^{**}(w_i)$. The reason lies in the fact that all gates represent a monotone function and that the on-set relation above holds separately for each gate. Q.E.D.

It is important to note that divergent (nonreconvergent) fanout branches do not have to be cut. These lines will get the signal probability of their ancestor during the computation process.

*The Restricted-Range Cutting Algorithm*

Here we show how to take advantage of the inversion parity of the reconvergent fan-out branches to compute tighter bounds. We show that in some cases it is possible to assign one of the ranges $[0, g]$ or $[g, 1]$ to the cut fan-out branch, where $g$ is the signal probability of the stem of the fan-out. If $g$ is in itself a range $[l, u]$, then one of the ranges $[0, u]$ or $[l, 1]$ will be assigned.

*Theorem 3:* Let $w$ be a fan-out stem with signal probability $g$, from which only one pair of reconverging paths emanates. Let this pair of reconverging paths be the only one in the network. Let $H$ be the gate through which the pair of paths reconverges. Let $h$ be the output of this gate. Then, if a fan-out branch is cut and assigned a range according to the appropriate case of Table I, then all the signal probability ranges computed on the resultant tree will include the true values.

Before we prove Theorem 3, we will show an example.

*Example 2:* Consider the circuit of Fig. 6. The two paths emanating at $w$ and reconverging at $h$ have unequal inversion parities, and the type of the reconverging gate is a NOR. Suppose we would like to cut the upper path, which has an odd parity between $w$ and $h$. According to Table I, the assigned range should be [3/4, 1]. The signal probability ranges computed for the resultant tree are shown in Fig. 6.

*Proof of Theorem 3:* For the purpose of this proof, let $g$ be the Boolean function realized by the fan-out stem. We will prove one of the entries of Table I. The proof for the rest is similar. We prove it for the case where both paths have even parity and where the reconverging gate is an AND. We have to show that a range of $[P_s(g), 1]$ for the cut branch will yield true bounds on all other lines in the circuit. Because of the assumptions imposed in the theorem, it is sufficient to prove that the signal probability range computed for line $h$ includes the true value. The Boolean function of $h$ expressed in terms of $g$ can be written as

$$h = (A_1 g + B_1)(A_2 g + B_2)$$

where $g$ is independent of $A_1, B_1, A_2, B_2$.

If the first fan-out branch is cut and assigned a primary input $x$, then the function realized by $h$ expressed in terms of $x$ and $g$ is

$$h^* = (A_1 x + B_1)(A_2 g + B_2).$$

The question now is what should $p_s(x)$ be so that

$$p_s(h) = p_s(h^*). \tag{3}$$

The computation of $p_s(h)$ and $p_s(h^*)$ yields

$$p_s(h) = p_s[(A_1A_2\overline{B}_1 + A_1\overline{B}_1B_2 + A_2B_1\overline{B}_2)g] + p_s(B_1B_2) \tag{4}$$

TABLE I
THE RESTRICTED RANGES ASSIGNED TO A CUT FAN-OUT BRANCH AS A FUNCTION OF TYPE OF RECONVERGING GATE, THE PATH INVERSION PARITY, AND THE SIGNAL PROBABILITY OF THE STEM

| PARITY OF CUT BRANCH BETWEEN $w$ AND $h$ | | TYPE OF RECONVERGING GATE, H: AND/NOR | TYPE OF RECONVERGING GATE, H: OR/NAND | INVERSION PARITIES OF RECONVERGING PATHS |
|---|---|---|---|---|
| EVEN | | $[g,1]$ | $[0,g]$ | EQUAL |
| ODD | | $[0,g]$ | $[g,1]$ | |
| EVEN | | $[0,g]$ | $[g,1]$ | UNEQUAL |
| ODD | | $[g,1]$ | $[0,g]$ | |

$$p_s(h^*) = p_s[A_1\overline{B}_1(A_2g + B_2)]p_s(x) + p_s(A_2B_1\overline{B}_2g) + p_s(B_1B_2). \tag{5}$$

Substitution of (4) and (5) into (3) results in

$$p_s(x) = \frac{p_s[A_1\overline{B}_1(A_2 + B_2)g]}{p_s[A_1\overline{B}_1(A_2g + B_2)]}. \tag{6}$$

Note that $g = 0 \rightarrow p_s(x) = 0$, and $g = 1 \rightarrow p_s(x) = 1$.

Moreover, note that

$$p_s(x) - p_s(g) = \frac{p_s(A_1A_2\overline{B}_1\overline{B}_2\overline{g})p_s(g)}{p_s[A_1\overline{B}_1(A_2g + B_2)]} \geq 0,$$

which completes the proof. Q.E.D.

The restricted range cutting algorithm is described in the following procedure.

*Procedure 2:*

*Step 1:* Assign a signal probability of $p_i = 1/2$ to all input lines, and compute the signal probabilities of all the tree lines.

*Step 2:* Moving from the inputs towards the outputs, cut reconvergent fan-out branches to turn the circuit into a tree. When a fanout branch is cut, assign to it a restricted range, if all reconverging pairs of paths in which this line participates, along with their corresponding reconverging gates, yield the same restricted range according to Table I; otherwise assign a full-range to the cut branch.

*Step 3:* Propagate the bounds on the resultant tree.

*Example 3:* Consider the circuit of Fig. 5, redrawn in Fig. 7. The tree line values were computed in Example 1. In Fig. 7(a) we show the computation of $p_s(h_1)$ and $p_s(h_2)$ based on restricted ranges. Suppose we want to cut line $w_2$. Line $w_2$ participates in two pairs of reconverging paths: the pair $\{G_1G_4G_5, G_2G_5\}$ with $G_5$ the reconverging gate; and the pair $\{G_1G_4G_5G_6, G_3G_6\}$ with $G_6$ the reconverging gate. In the first pair, $w_2$ lies on an odd path where the paths have unequal inversion parities. In the second pair, $w_2$ lies on an even path where the paths have equal inversion parities. According to Table I, in both these two cases, the range to be assigned to line $w_2$ should be $[0, g]$, where $g$ is the signal probability of the stem, $w_1$. Thus, in Fig. 7(a), a range of [0, 3/4] is assigned to line $w_2$. After we cut line $w_2$, there is only one pair of reconverging paths left, namely, $\{G_2G_5G_6, G_3G_6\}$. Suppose we want to cut line $w_4$, next. This is the case where the paths have unequal inversion parities, and where $w_4$ lies on the even path. According to Table I, the range to be assigned to it is $[g, 1] = [3/4, 1]$. Fig. 7(a) shows the propagation of the bounds to the

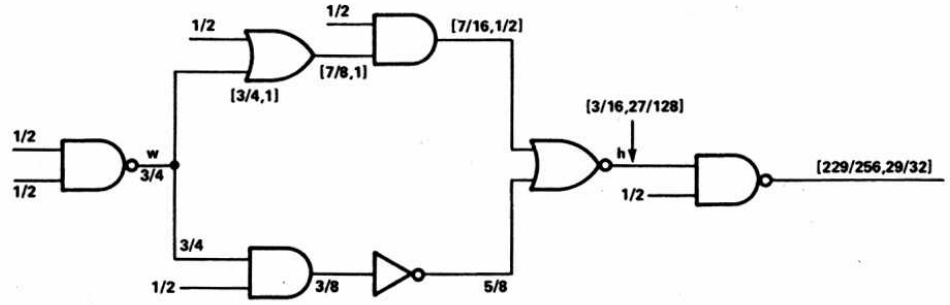


Fig. 6. Example 2.

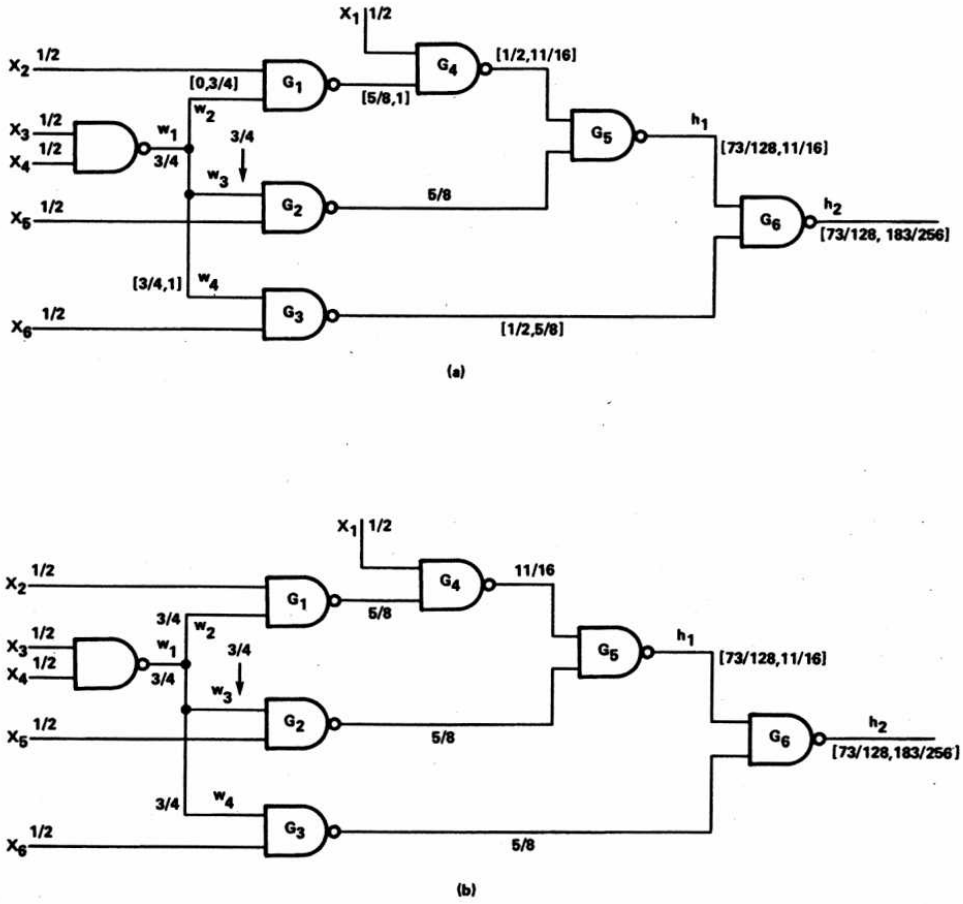


Fig. 7. (a) Nontree bounds computed with restricted ranges. (b) The final signal probability bounds.

nontree lines, $h_1$ and $h_2$. Fig. 7(b) shows the signal probability bounds for all lines in the circuit.

Note that Procedure 2, Step 2 calls for cutting fan-out branches as you move from inputs to outputs. Thus, when a line $w$ is considered for cutting, there should be no stem of fan-out in $c(w)$. In other words, all fanout lines prior to line $w$ should be resolved by the time line $w$ is considered.

Note also that there is room for optimizing the bounds by properly choosing the cut branches. For example, if a fan-out stem, $g$, has signal probability larger than $1/2$ and the branches following it are such that if one is cut a bound $[0, g]$ should be assigned; and if the other one is cut, a bound $[g, 1]$ should be assigned, then, at least locally, the second choice would result in a tighter bound. Note also that if we had decided to cut line $w_4$ before cutting line $w_2$, we would have had to assign the full range to it. This would have probably resulted in a looser bound for lines $h_1$ and $h_2$.

*Theorem 4:* The restricted-range cutting algorithm computes true bounds.

*Proof:* As outlined in Procedure 2, fan-out branches are cut starting from the inputs. Thus, when a fan-out branch has to be cut, there is no fan-out stem left in its cone of influence (as required by Theorem 3). If according to all possible pairs of reconverging paths in which this fan-out branch participates, the required restricted range is unique, then such an assignment will result in signal probability bounds that enclose the true value for each and every line in the circuit (if computed). If, however, according to Table I, one pair of paths requires one type of restricted range, while another pair of paths requires the other one, Procedure 2 would assign to that branch the full range, which would also result in true bounds, according to Theorem 2. Therefore, since the network is turned out into a tree sequentially, and since each time a branch is cut its attached range is valid, then the final end result is that the

computed bounds will enclose the true signal probability for each and every line in the circuit. Q.E.D.

## HEURISTICS

Although the cutting algorithm is our engine for computing signal probabilities, it is worthwhile to also have a number of heuristics that we can switch to when tighter bounds are required. Some insight into the importance of the ordering of the cuts was given in the previous section; this section will continue in that vein. In the following heuristics we show how to improve the signal probability bounds based on multiple runs of the cutting algorithm.

*Lemma:* If the cutting algorithm is run $k \geq 1$ times and if the signal probability of a wire is computed to be in the range $[l_i, u_i]$ for the $i$th trial, then the signal probability of the wire is bounded by $[l, u]$ where

$$l = \operatorname*{Max}_i \{l_i\}, i = 1, 2, \cdots, k \tag{7}$$

$$u = \operatorname*{Min}_i \{u_i\}, i = 1, 2, \cdots, k. \tag{8}$$

*Proof:* Suppose the cutting algorithm is applied $k$ times, each time recording the computed bounds for each line in the circuit. Let $w$ be a line in the circuit and let the computed bounds for this line be $[l_i, u_i], i = 1, 2, \cdots, k$. Then, since each bound constitutes a true range, the combined bound for this line may be set to $[l, u]$, according to (7) and (8). Q.E.D.

### *Coin Flipping*

Coin flipping is an *a priori* unbiased way of cutting fan-out branches. Suppose $w$ is a fan-out stem of a $q$-way reconverging fanout. In any assignment of ranges, one fan-out branch will be assigned the signal probability of the stem, $p_s(w)$, while all the rest will be cut and assigned appropriate bounds. Obviously, there are $q$ ways in which the line assigned to $p_s(w)$ can be selected. The coin flipping method would flip a fair $q$-way coin to decide which line would be selected. Note that if only full ranges are considered, there are $q$ different cutting patterns associated with this $q$-way fan-out. If, however, restricted ranges are used, more patterns are possible.

Suppose the coin flipping strategy is applied for $k$ times. Then, based on the results of these $k$ trials, a tight bound can be computed for each line in the circuit by using (7) and (8).

### *Group Exhaustion*

In this heuristic we group the fan-out stems in small groups to make it possible to exhaust all cutting patterns within a group and compute tight bounds based on (7) and (8). It is worthwhile to group fan-out stems that appear "close" in terms of circuit connectivity.

### *Adaptive Cuts*

Suppose that after computing the bounds based on some cutting patterns, some lines still have unacceptable ranges. In the next cutting pattern it is possible to orient the cuts in such a way that true values (or tight bounds) would be propagated towards those lines, thus improving the bounds.

### *Macros*

Many regular structures constitute repetitions of a basic macro. A ripple carry adder repeats the basic structure of a full adder. The number of repetitions depends on the width of the memory word. An EXCLUSIVE-OR gate is used regularly in many digital designs. If an input/output signal probability bound relation were established for these macros, considerable CPU time could be gained when they are used in the circuit.

Consider a two-input EXCLUSIVE-OR gate. Let the signal probability bounds on the inputs be $[l_1, u_1]$ and $[l_2, u_2]$. Let the output signal probability bound be $[l, u]$. Based on the full-range assignment, the following input/output relations can be derived.

If the cones of influence of the two inputs to the gate are disjoint, then it can be shown that

$$l = \operatorname*{Min}_i \{s_i\} \tag{9}$$

$$u = \operatorname*{Max}_i \{s_i\} \tag{10}$$

where

$$\begin{aligned} s_1 &= l_1 + l_2 - 2l_1l_2 \\ s_2 &= l_1 + u_2 - 2l_1u_2 \\ s_3 &= u_1 + l_2 - 2u_1l_2 \\ s_4 &= u_1 + u_2 - 2u_1u_2. \end{aligned}$$

However, if the cones of influence of the two inputs to the gate are conjoint, then the relationship can be shown to be

$$l = \operatorname{Max}\{l_1(1 - u_2), l_2(1 - u_1)\} \tag{11}$$

$$u = \operatorname{Min}\{1 - l_1l_2, u_1 + u_2 - u_1u_2\}. \tag{12}$$

Consider now an iterative network like the one shown in Fig. 8. Assume that all the fan-out stems and reconverging paths are "hidden" within the blocks. In other words, assume that there are no reconverging paths crossing block boundaries. In this case, it is possible to treat each block as a macro and propagate bounds from inputs to internal macro lines, as well as to macro outputs, by using transfer relations.

A ripple carry-adder satisfies the requirement above. In this case each block (except for the first) is a full-adder stage receiving three inputs (two external inputs and a carry from the previous stage) and generating two outputs (the sum bit and the carry to the next stage). A full-adder stage is shown in Fig. 9. Given that the signal probabilities to lines $A_i$, $B_i$, and $C_{i-1}$ are respectively $[l_1, u_1]$, $[l_2, u_2]$, and $[l_3, u_3]$, the full-range cutting algorithm yields the following bounds for the other macro lines:

$$l_4 = l_1l_2, u_4 = u_1u_2 \tag{13}$$

$$l_5 = l_1 + l_2 - l_1l_2, u_5 = u_1 + u_2 - u_1u_2 \tag{14}$$

$$l_6 = l_3(l_1 + l_2 - l_1l_2), u_6 = u_3(u_1 + u_2 - u_1u_2) \tag{15}$$

$$l_7 = l_1l_2l_3, u_7 = u_1u_2u_3 \tag{16}$$

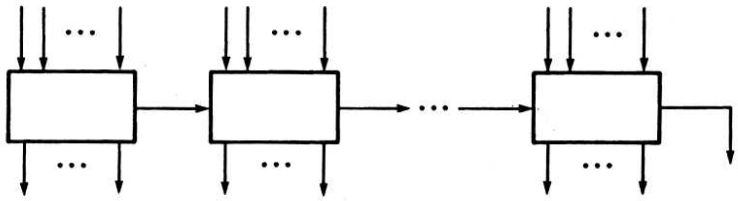

Fig. 8. An iterative network with no cross-boundary fan-out.

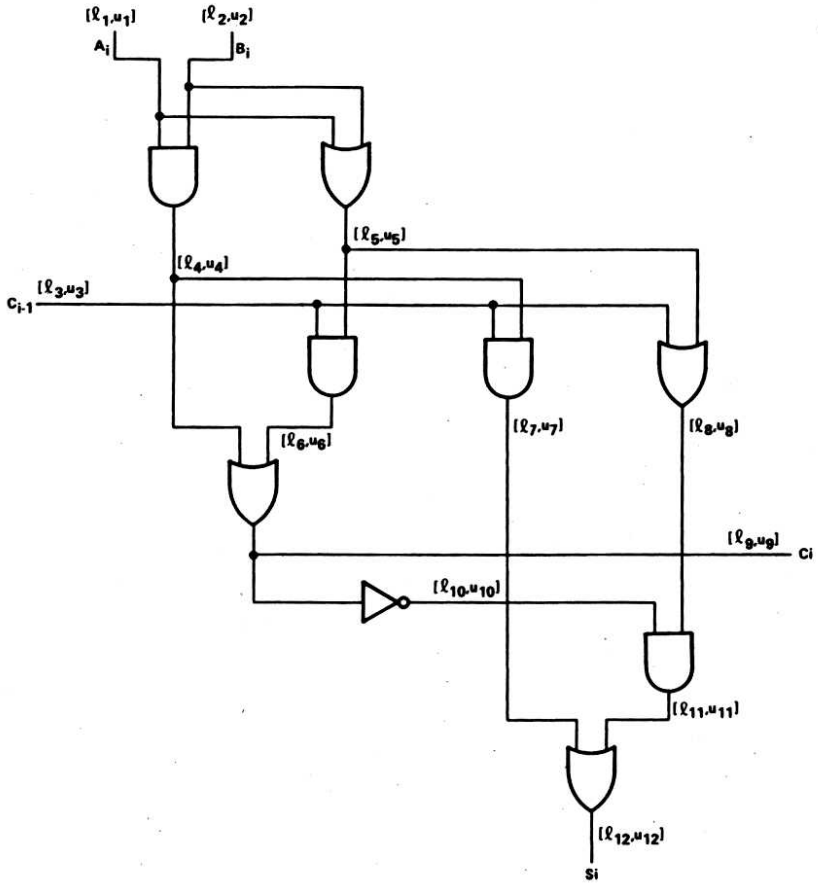


Fig. 9. A full adder stage.

$$l_8 = 1 - (1 - l_1)(1 - l_2)(1 - l_3),$$
$$u_8 = 1 - (1 - u_1)(1 - u_2)(1 - u_3) \quad (17)$$
$$l_9 = \text{Max}\{l_1 l_2, l_3(l_1 + l_2 - l_1 l_2)\},$$
$$u_9 = u_3 + u_1 u_2 (1 - u_3) \quad (18)$$
$$l_{10} = 1 - u_9, u_{10} = 1 - l_9 \quad (19)$$
$$l_{11} = \text{Max}\{l_1(1 - u_2)(1 - u_3), l_2(1 - u_1)(1 - u_3)\},$$
$$u_{11} = \text{Min}\{1 - l_1 l_2, 1 - l_2 l_3, 1 - l_1 l_3\} \quad (20)$$
$$l_{12} = l_1 l_2 l_3, u_{12} = \text{Min}\{1 - (1 - u_1) l_2 l_3,$$
$$1 - (1 - u_2) l_1 l_3, 1 - (1 - u_3) l_1 l_2\}. \quad (21)$$

By placing proper initial conditions on the primary inputs, all the signal probability bounds in the network can be computed very fast (by a computer program).

*Expansion*

It is possible to use the cutting algorithm jointly with Shannon's expansion to achieve both tighter bounds and gain in CPU time. Suppose a signal probability range is computed for line $w_1$. Suppose, further, that line $w_2$, which is reachable from $w_1$ (namely, $w_1 \in c(w_2)$), does not have an acceptable signal probability range. It is possible to improve the bound on line $w_2$ by computing the signal probabilities of two special cases. Let $p_s(w_2/w_1 = 0)$ ($p_s(w_2/w_1 = 1)$) be the conditional signal probability for line $w_2$ given that line $w_1$ is fixed at zero (fixed at one). Then, the following theorem holds.

*Theorem 5:* If $p_s(w_2/w_1 = 0) \in [a_0, a_1]$,

$$p_s(w_2/w_1 = 1) \in [b_0, b_1],$$

and $p_s(w_1) \in [c_0, c_1]$, then $p_s(w_2) \in [d_0, d_1]$ where

$$d_0 = a_0(1 - c_1) + b_0 c_0 \quad (22)$$
$$d_1 = \text{Min}\{a_1(1 - c_0) + b_1 c_1, 1\}. \quad (23)$$

*Proof:* By Shannon's expansion we have

$$p_s(w_2) = p_s(w_2/w_1 = 0) \cdot [1 - p_s(w_1)] + p_s(w_2/w_1 = 1) \cdot p_s(w_1).$$

Substituting the ranges in this formula yields

$$[d_0, d_1] = [a_0, a_1] \cdot [1 - c_1, 1 - c_0] + [b_0, b_1] \cdot [c_0, c_1].$$

Equating lower bounds of both sides of the equation yields

$$d_0 = a_0(1 - c_1) + b_0c_0.$$

Equating upper bounds of both sides of the equation yields

$$d_1 = a_1(1 - c_0) + b_1c_1.$$

However, since the right-hand side of this equation may be larger than one, in general, and since the left-hand side should always be smaller than one, we have to take

$$d_1 = \text{Min}\{a_1(1 - c_0) + b_1c_1, 1\}. \qquad \text{Q.E.D.}$$

The following corollary shows a useful application of Theorem 5.

*Corollary:* Let the gate $G_2$ have the output $w_2$. If there is no fan-out stem in $c(w_1)$ whose reconvergent gate is $G_2$, then $p_s(w_2/w_1 = 0)$ $(p_s(w_2/w_1 = 1))$ can be computed by simply considering the residual circuit resulting after cutting line $w_1$ and fixing its input counterpart to a fixed 0 (1).

*Proof:* If the restriction imposed by the corollary holds, then $p_s(w_2/w_1 = 0)$ $(p_s(w_2/w_1 = 1))$ are independent of the primary inputs in $c(w_1)$. Q.E.D.

*Example 4:* Consider the circuit of Fig. 10(a). In this example we will show how expansion around line $w$ is used to compute the signal probability bound on the output line $F$. Note that the restriction of the corollary holds for lines $w$ and $F$. The following bounds have been computed by using the restricted-range cutting algorithm.

$$p_s(w) \in [3/8, 39/64].$$

Forcing $w$ to a zero yields $F = 1$. Thus

$$p_s(F/w = 0) \in [1, 1].$$

Forcing $w$ to a one yields the circuit of Fig. 10(b). The signal probability bound on $F$ is computed to be

$$p_s(F/w = 1) \in [1/8, 15/64].$$

Thus,

$$p_s(F) \in [7/16, 3145/4096].$$

Note that the usefulness of the expansion heuristic rests on the observation that it is easier to compute bounds for the two special cases than for the original circuit. This is because the circuit resulting after forcing $w_1$ to either zero or to one is simpler than the original circuit.

## Computation of Lower Bounds for Detection Probabilities

The algorithm is spelled out in the following procedure:

*Procedure 3:*

*Step 1:* Pick a prime fault which has not been considered yet. If all prime faults considered—stop.

*Step 2:* Choose a single[1] propagation path, that has not yet been considered, from the site of the fault to a primary output. If all single paths have been considered, mark the fault as "hard-fault" and go to Step 1.

*Step 3:* Introduce the auxiliary AND gate whose signal probability equals the detection probability along the selected path.

*Step 4:* Compute the lower bound of the signal probability on the output of the auxiliary gate. If the lower bound is acceptable (larger than the threshold), mark it off as an "easy-fault" and go to Step 1. Otherwise, go to Step 2.

Note that only single propagation paths are treated in Procedure 3. A fault that can be detected only along *multiple paths* will be marked as hard.

*Example 5:* Consider the circuit of Fig. 11. In Fig. 11(b) we show the computation of a lower bound for $x_3/0$. In Fig. 11(c) we show the computation of a lower bound for $x_4/1$. Notice that we propagate the fault $x_3/0$ along a single path, and the fault $x_4/1$ along either a single or a double path. Table II displays the detection probabilities and the computed lower bounds for all the prime faults in the circuit. The exact detection probability figures were computed by the Boolean difference technique, and the lower bounds by the cutting algorithm.

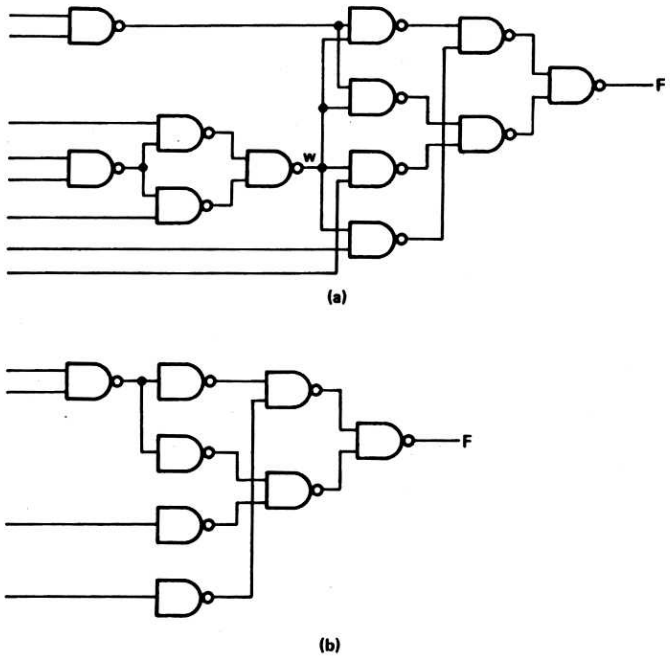


Fig. 10. (a) The circuit of Example 4. (b) The circuit resulting after forcing $w$ to a one.

## Correction of Hard Faults

Once the hard faults are identified it is necessary to upgrade their detection probabilities so that they will be susceptible to random patterns. Some hard faults are known to be introduced by high fan-in gates. The probability of detecting a stuck-at-zero fault at the input of an $n$-way AND is $2^{-n}$. A similar situation occurs with other high fan-in gates, like OR, NAND, and NOR. Faults at the inputs of a long cascade of alternating AND's and OR's are also hard to detect. Faults in PLA's realizing large product terms are also hard to detect. Hard faults may appear also in an unstructured design. All it takes for a fault to be hard [20] is to have a small cardinality of inter-

[1] In the case where all propagation paths from the site of the fault to a primary output have equal inversion parity, this requirement can be relaxed.

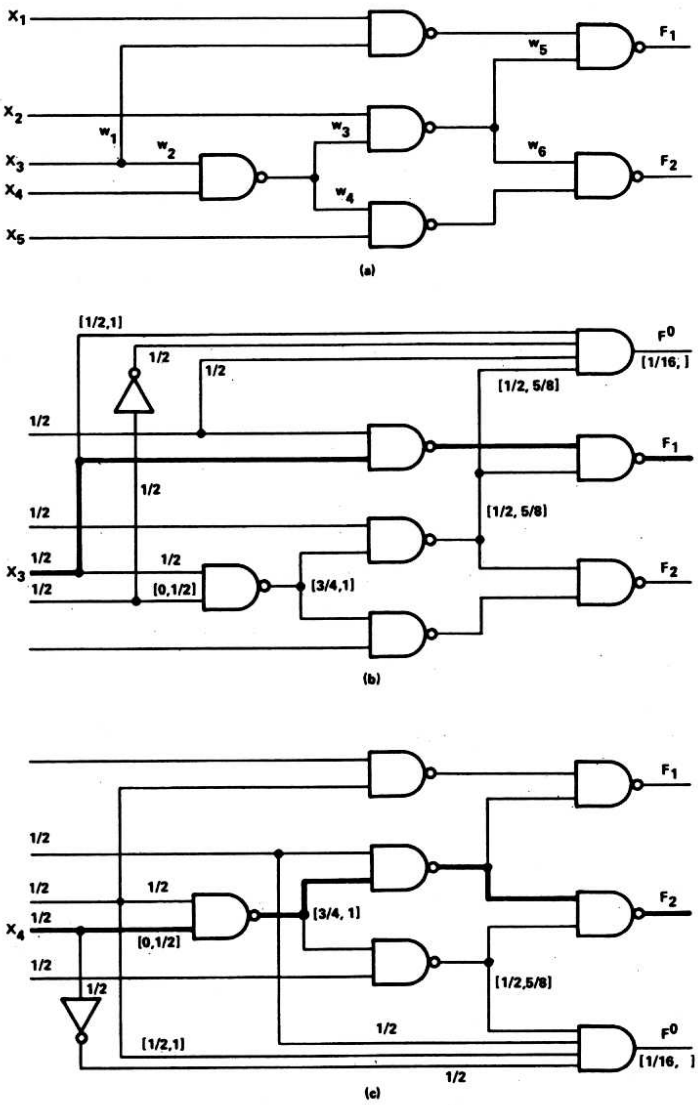


Fig. 11. (a) The circuit of example 5. (b) Computation of the lower bound of the detection probability of $x_3/0$ along the bold face line. (c) The same as (b), but for $x_4/1$.

TABLE II
THE DETECTION PROBABILITY, $p_d$, AND THE COMPUTED LOWER BOUND OF THE DETECTION PROBABILITY FOR ALL PRIME FAULTS IN EXAMPLE 5

| Prime Fault | $p_d$ | Lower Bound |
|---|---|---|
| $x_1/0$ | 3/16 | 5/32 |
| $x_1/1$ | 3/16 | 5/32 |
| $x_2/0$ | 11/32 | 9/32 |
| $x_2/1$ | 11/32 | 9/32 |
| $x_3/0$ | 9/32 | 1/16 |
| $x_3/1$ | 9/32 | 1/16 |
| $x_4/0$ | 3/16 | 5/64 |
| $x_4/1$ | 3/16 | 1/16 |
| $x_5/0$ | 3/16 | 3/16 |
| $x_5/1$ | 3/16 | 3/16 |
| $w_1/0$ | 3/16 | 5/32 |
| $w_1/1$ | 1/8 | 1/8 |
| $w_2/0$ | 3/16 | 5/64 |
| $w_2/1$ | 3/16 | 1/16 |
| $w_3/0$ | 11/32 | 3/16 |
| $w_3/1$ | 1/8 | 5/64 |
| $w_4/0$ | 3/16 | 3/16 |
| $w_4/1$ | 1/8 | 5/64 |
| $w_5/0$ | 7/16 | 3/8 |
| $w_5/1$ | 5/16 | 9/32 |
| $w_6/0$ | 7/16 | 25/64 |
| $w_6/1$ | 3/16 | 3/16 |

section between the controllability and observability sets. So, in general, hard faults may be randomly spread in a logic design. The cutting algorithm serves as an engine to identify these faults. Some thoughts on random pattern design for testability are reported in [21]. Here we present some of our thoughts on the subject.

The first approach, a unified one, is general enough to be applied to any design. The idea is to use a semirandom rather than a completely random self test. After the hard faults have been identified, compute a set of tests that cover these faults and store them in a read-only memory (ROM) built on chip. All the experiments conducted until now support the conjecture that there is not a large number of hard faults. So, with a relatively small area overhead, all the desired test patterns for hard faults can be stored on chip. The self test therefore would consist of two parts. In the first, all the hard faults would be exercised with the deterministic patterns stored in the ROM, and the signature accumulated in the MISR. In the second part, the input stimulus would be switched to a random mode. Random patterns are continuously applied to the circuit while the signature keeps accumulating in the MISR. The final signature for the chip is the combined signature of the two test modes, and this is the one to use as signature reference. The advantage of this approach is that it is unified, in the sense that it can be readily applied to any design. The disadvantage is that it uses two test modes.

The second approach is circuit-dependent. Once a hard fault has been identified, it is possible to ease its testing by random inputs by introducing control lines and test points along the propagation path of the fault. It is also possible to introduce new primary outputs to make the fault more observable. Note that in an LSSD environment one can implement these extra inputs and outputs by introducing extra latches in the SRL chains rather than by actual I/O pins, which may be scarce. The advantage of this approach is that the self test will have only one mode—the random mode—rather than both the deterministic and the random that the previous approach had. The disadvantage is that the design for testability is circuit-dependent and has to be treated case by case. It is still to be found which approach leads to smallest hardware overhead. Our belief is that the hardware overhead will not exceed 5 percent of the chip area, including both the hardware modifications necessary for self test due to the MISR's and the overhead penalty due to random pattern resistant faults. For more information on correction of hard faults consult [22].

### *Application to Other Disciplines*

The cutting algorithm could also be used in other disciplines, like design verification, current-switching problems (known as $\Delta I$) [23]–[25], timing analysis, etc.

The problem one faces in design verification is whether a design meets its objective function. Put in different terms, suppose it is necessary to verify that design number 1 is logically equivalent to design number 2. One design may be human-created, the other generated automatically. Or, one design may have been produced by a logic designer; the other, by simulation.

Suppose we take the two designs and EXCLUSIVE-OR their corresponding outputs, namely, output 1 of design 1 with

output 1 of design 2, etc. Suppose also that the corresponding inputs of the designs are tied together, namely, input 1 of design 1 to input 1 of design 2, etc. Then obviously, if the two designs are logically equivalent, the signal probability of the outputs of the XOR gates should be zero.

The cutting algorithm, if run, can indicate whether the two designs are different. If the signal probability range is computed for each XOR output, and if at least one of the ranges discovers a nonzero lower bound, then certainly the two designs differ. If all lower bounds are zero, no conclusion can be drawn. Notice that this scheme may help to screen out bad designs, but it will fail, as other methods do also, to tell for certain that the two designs are identical. To confirm that point requires exhaustion of their input space to see whether a specific input results in a different output.

A current-switching problem may occur when more than a specified number of outputs simultaneously change value, due to an input change. To see whether there is a $\Delta I$ problem, we construct the network of Fig. 12.

The outputs of two copies of the circuit are XORed respectively. The outputs of the XOR gates feed a threshold detection circuit, whose output is $F = 1$ if and only if more than $k$, say, inputs are one. The inputs to the two copies of the circuit are considered independent inputs. The input vectors to the two copies should be considered as two contiguous input vectors applied to the original circuit. Then if $F = 1$, we have discovered a $\Delta I$ problem due to an input change.

The cutting algorithm can be used to compute a bound on the probability of a $\Delta I$ problem with parameter $k$ by computing the signal probability of the output $F$. This bound will tell how likely it is to have a $\Delta I$ problem in the original design.

Notice that this procedure, in fact, underestimates slightly the desired probability. In the computation model of Fig. 12, we have not excluded the possibility that the two input vectors to the two copies are identical. Thus, the cardinality of the input space is $4^n$, rather than $4^n - 2^n$, where $n$ is the number of inputs to the circuit. However, even for small values of $n$ ($\geq 5$), the underestimation error is negligible.

A timing analysis program attempts to identify short paths and long paths of propagation in a logic design. Since the identification of these paths is based on delay equations, there is no guarantee that the paths marked off by the program as either short or long are at all functional (exercisable). Once a potentially long (or short) path is identified, it is possible to test whether it is functional by the cutting algorithm. As in the detection probability calculation method, it is possible to display the conditions necessary to activate the path by an auxiliary AND gate and compute the signal probability at its output. If the lower bound of this signal probability is nonzero, then the path is functional; otherwise, no conclusion can be drawn.

## Summary and Conclusions

The problem of random pattern testability, in the context of self test, was discussed in this paper. The objective was to determine which faults, if any, manifest resistance to random pattern stimuli. We have shown how the cutting algorithm can be used to identify these hard faults in any design. Once these hard faults are identified it is possible to make them easy by paying some hardware penalty either in a ROM, or in a close proximity to the site of the hard fault. We have also discussed some heuristics that should make the cutting algorithm efficient and practical.

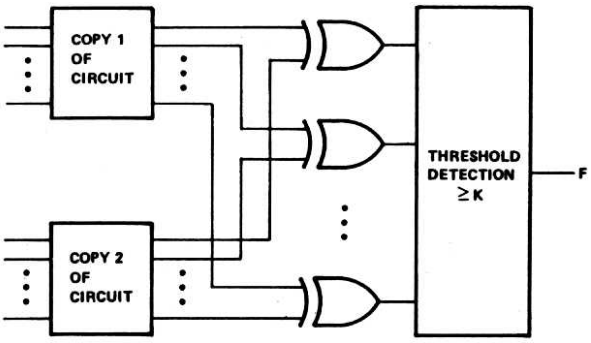


Fig. 12. Detection of a $\Delta I$ problem in a logic design.

The question arises as to what the complexity is of this new approach. A careful study is currently being conducted and will be reported in [26]. A rough estimate can be given now. If $N$ is the number of gates in the circuit, then in the worst case the complexities are:

| | | |
|---|---|---|
| Computing signal probabilities | with full ranges: | $O(N^2)$ |
| | with restricted ranges: | $O(N^3)$ |
| Computing detection probabilities | with full ranges: | $O(N^2)$ |
| | with restricted ranges | $O(N^3)$. |

These are worst-case mathematical bounds, and we expect the actuality to be much less.

## References

[1] O. H. Ibara and S. K. Sahni, "Polynomially complete fault detection problems," *IEEE Trans. Comput.*, vol. C-24, pp. 242–249, Mar. 1975.

[2] H. Fujiwara and S. Toida, "The complexity of fault detection: An approach to design for testability," in *Proc. 12th Annu. Fault-Tolerant Comput. Symp.*, June 1982, pp. 101–108.

[3] P. Goel, "Test generation costs analysis and projections," in *Proc. 17th Design Automat. Conf.*, June 1980, pp. 79–84.

[4] J. Savir, "Syndrome-testable design of combinational circuits," *IEEE Trans. Comput.*, vol. C-29, pp. 442–451, June 1980; and "Correction to 'Syndrome-testable design of combinational circuits,'" *IEEE Trans. Comput.*, vol. C-29, pp. 1012–1013, Nov. 1980.

[5] Z. Barzilai, J. Savir, G. Markowsky, and M. G. Smith, "The weighted syndrome sums approach to VLSI testing," *IEEE Trans. Comput.*, vol. C-30, pp. 996–1000, Dec. 1981.

[6] A. K. Susskind, "Testing by verifying Walsh coefficients," in *Proc. 11th Annu. Fault-Tolerant Comput. Symp.*, June 1981, pp. 206–208.

[7] J. C. Muzio and D. M. Miller, "Spectral techniques for fault detection," in *Proc. 12th Annu. Fault-Tolerant Comput. Symp.*, June 1982, pp. 297–302.

[8] E. J. McCluskey and Bozorgui-Nesbat, "Design for autonomous test," *IEEE Trans. Comput.*, vol. C-30, pp. 866–875, Nov. 1981.

[9] P. H. Bardell and W. H. McAnney, "Self-testing of multichip logic modules," in *Proc. 1982 Int. Test Conf.*, Nov. 1982, pp. 200–204.

[10] B. Konemann, J. Mucha, and G. Zwiehoff, "Built-in logic block observation techniques," in *Proc. 1979 Int. Test Conf.*, Oct. 1979, pp. 37–41.

[11] P. P. Fasang, "BIDCO, built-in digital circuit observer," in *Proc. 1980 Int. Test Conf.*, Nov. 1980, pp. 261–266.

[12] J. Mucha, "Hardware techniques for testing VLSI circuits based on built-in test," in *Proc. COMPCON 81*, Feb. 1981, pp. 366–369.

[13] E. B. Eichelberger and T. W. Williams, "A logic design structure for LSI testability," in *Proc. 14th Design Automat. Conf.*, June 1977, pp. 462–468.

[14] K. P. Parker and E. J. McCluskey, "Analysis of logic circuits with faults

using input signal probabilities," *IEEE Trans. Comput.*, vol. C-24, pp. 573-578, May 1975.
[15] ——, "Probabilistic treatment of general combinational networks," *IEEE Trans. Comput.*, vol. C-24, pp. 668-670, June 1975.
[16] I. Koren, "Analysis of the signal reliability measure and an evaluation procedure," *IEEE Trans. Comput.*, vol. C-28, pp. 244-249, Mar. 1979.
[17] K. K. Aggarwal, "Output probability expression for general combinational networks," *Microelectron. Reliab.*, vol. 17, pp. 601-602, 1978.
[18] P. Agrawal and V. D. Agrawal, "Probabilistic analysis of random test generation method for irredundant combinational logic networks," *IEEE Trans. Comput.*, vol. C-24, pp. 691-695, July 1975.
[19] M. A. Breuer and A. D. Friedman, *Diagnosis and Reliable Design of Digital Systems*. Silver Spring, MD: Computer Science Press, 1976.
[20] J. Savir, "Good controllability and observability do not guarantee good testability," IBM Res. Rep., RC 9432, June 1982.
[21] K. L. Meinert, "Designing for testability with GUEST," *Hewlett-Packard J.*, p. 28, Mar. 1982.
[22] E. B. Eichelberger and E. Lindbloom, "Random-pattern coverage enhancement and diagnosis for LSSD logic self-test," *IBM J. Res. Develop.*, vol. 27, pp. 265-272, May 1983.
[23] E. E. Davidson, "Electrical design of high-speed computer package," *IBM J. Res. Develop.*, vol. 26, pp. 349-361, May 1982.
[24] A. Brown and G. Ditlow, "Delta-I failure detection technique," *IBM Tech. Disclosure Bulletin*, to be published.
[25] G. Ditlow, J. Savir, and A. Brown, "Computing the probability of a Delta-I failure," *IBM Tech. Disclosure Bulletin*, to be published.
[26] T. H. Spencer, "Complexity analysis of the cutting algorithm," in preparation.

**Jacob Savir** (S'76-M'78) received the B.Sc. (cum laude) and M.Sc. degrees in electrical engineering from the Technion—Israel Institute of Technology, Haifa, Israel, in 1968 and 1973, and the M.S. in statistics and Ph.D. in electrical engineering from Stanford University, Stanford, CA, in 1976 and 1977.

Since 1967 he has held several technical and teaching positions. In the technical area he served in the Israel Defense Forces from 1968-72; worked for Israel Ministry of Defense from 1974-75; worked with Prof. E. J. McCluskey, as a research assistant, and as a post doctoral fellow, on intermittent fault problems, and syndrome testing from 1975-78. He joined IBM in 1978 at the T. J. Watson Research Center, as a research staff member, to work on testing problems in VLSI. He is currently on sabbatical at the IBM Data Systems Division in Poughkeepsie, NY. He served on several conference committees, including FTCS-11, and FTCS-12, and published numerous papers in the testing field. In the academic area he has taught numerous courses in mathematics, physics, statistics, electrical engineering, and computer science at the Technion—Israel Institute of Technology, in 1967 and from 1972-75, and at Ort Singalovsky, Tel-Aviv, from 1969-1974. Since 1979 he has been an Adjunct Professor at Pace University, New York, teaching undergraduate, and graduate courses, at the Department of Math and Sciences, and the Department of Computer and Information Systems. His research interests include test generation, self-test, design automation, design for testability, data compression for testing purposes, and related topics.

Dr. Savir is a member Sigma Xi.

**Gary S. Ditlow** (M'82) was born in Lancaster, PA on May 24, 1949. He received the B.S.E.E. degree in 1971 from the University of Maryland and the M.S. in computer science from Rensselaer Polytechnic Institute in 1973.

In 1973 he joined the IBM Poughkeepsie Laboratory where he developed algorithms in the areas of chip placement, package analysis, graphics, design verification, and binary trees for data base design. Later he moved to IBM Fishkill and designed bipolar chips and circuits in the VLSI exploratory circuits area. In 1982, he joined the staff of the IBM T. J. Watson Research Center where his current interests are random pattern testability, circuit analysis using waveform relaxation, and the creation of logic equations for VLSI FET circuits.

Mr. Ditlow is a member of Tau Beta Pi and Eta Kappa Nu.

**Paul H. Bardell** (S'53-M'57-SM'71) received the B.S.E.E. from the University of Colorado, Boulder, CO, and the M.S.E.E. and Ph.D. from Stanford University, Stanford, CA.

Since joining IBM, he has worked in the areas of superconductors, semiconductor physics and integrated circuit design, especially semiconductor memory, and the test of complex circuit packages. He is currently Advanced Engineering in Manufacturing Manager for DSD, Poughkeepsie. His current interest is the testing of large digital networks.

Dr. Bardell is a member of the American Physical Society, the American Association for the Advancement of Science, the New York Academy of Science, and the Association for Computing Machinery.